The Dissertation Committee for Vinay Kiranshankar Siddavanahalli certifies that this is the approved version of the following dissertation:

Modeling and Visualization of Flexible Protein-protein Interactions

Committee:

Chandrajit Bajaj, Supervisor

Risto Miikkulainen

Arthur Olson

Gregory Plaxton

Vijaya Ramachandran

Peter Rossky

Modeling and Visualization of Flexible Protein-protein Interactions

by

Vinay Kiranshankar Siddavanahalli, B.S.,M.S.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

December 2006

This dissertation is dedicated to my parents and sister.

Acknowledgments

I was accepted as a Graduate Research Assistant under Dr. Chandrajit Bajaj in 2001. Ever since, it has been an amazing experience of collaborative research, discussions, and a few *Eureka* moments! It is my privilege to have had him as the guiding light of my research career for almost six years. I learnt many valuable lessons apart from computational geometry under his guidance, including: precision in expressing problems and solutions, correctness, completeness of research and the importance of writing and presenting research ideas.

Fourier Series and Transforms form a significant part of my research. Dr. Julio Castrillon, who was a Post Doc. under Dr. Bajaj, was responsible for teaching me all the nuances that go with complex abstract mathematics when it came to implementation. My research would have been unduly prolonged if not for his help in crucial junctures.

Our primary collaborators in the field of Molecular Biology include Dr. Arthur Olson and Dr. Michell Sanner from The Scripps Research Institute. They provided me with plenty of background knowledge and data from their field. My results were improved a lot after many video-conferences with them.

I obtained valuable support and ideas from all my fellow CVC group members. I would like to especially mention Dr Bong-soo Sohn (BongBong),

Jason Sun, Dr Zeyun Yu and Dr Jessica Zhang, who have been with me for more than five years. Anthony Thane proved to be a good programming partner and is currently enjoying in Microsoft.

There have been many friends in Austin, who have made my stay here a wonderful part of my life. Karlapalem Chandrashekar (KLC) and I have spent countless days and nights in the labs, breaking our heads on new research problems. Chui, Katherine and I spent long hours debating the environment, veganism and other *interesting* topics. I have had a multitude of training partners whom I have dragged along to the gym over the years, including Mitul, Simha, Suvrit and Prateek.

None of these accomplishments would have been possible without a lot of sacrifice from my parents.

Modeling and Visualization of Flexible Protein-protein Interactions

Publication No. _____

Vinay Kiranshankar Siddavanahalli, Ph.D.
The University of Texas at Austin, 2006

Supervisor: Chandrajit Bajaj

Protein-protein interactions form the basis of macromolecular formation and function. Determining a relative transformation for a pair of proteins and their conformations which form a stable complex, reproducible in nature, is known as protein-protein docking. Computational approaches to protein-protein docking are therefore a necessary pathway to virtual drug screening, plausible macro-molecular structures, and elucidating the function of proteins in assemblages. Protein conformational changes play a crucial role in such interactions, leading to a very high dimensional search space. The computational challenge is further increased as we obtain imaging data for larger and larger proteins, bridging the gap between proteins and cells. Traditional algorithms for the construction and visualization of protein structure and function have not scaled to handle large proteins, macromolecular assemblies and viruses.

In this thesis, we provide: data structures and algorithms to represent flexible protein structures, scalable error bounded techniques to compute soft protein-protein docking, a hierarchical flexible docking scheme and novel methods to visualize large interacting molecular complexes and assemblies.

Accurate and robust molecular surface computation is vital for parameterizing affinity functions and modeling interactions. We provide a adaptive grid based function definition, whose contours yield a family of relevant surfaces. We show that these are free of self intersections and provide methods to compute regions of C_0 continuity. The structure and functions of molecules are represented in a radial basis format, with smooth particle data representing electron density kernels, charges and solvent modulated dielectric coefficients. A fast summation algorithm, based on non-equispaced fast Fourier transforms, is presented to accurately, efficiently and adaptively compute these functions. Based on the previous surfaces and fast summation algorithms, we provide a model for soft docking and error-bounded approximation algorithms to solve the model and predict docking sites. The flexibility space is adaptively sampled using a domain decomposition of the protein into a Flexible Chain Complex. We then provide a flexible docking algorithm based on a multiresolution representation of the proteins, adaptive sampling of conformation, orientation spaces and greedy fit of residues at interfaces.

Scientific visualization of protein interfaces and active sites is employed for both data analysis and discovery. We provide algorithms to interactively render both the traditional ball and stick model of molecules and contours of

the sum of Gaussians based electron density. To visualize schematic models of large and flexible proteins at interactive rates and high quality, we introduce a novel hardware accelerated, imposter-based scheme to render curved surfaces like spherical patches, cylinders and helices, with correct per pixel shading, using limited geometric primitives. A telescoping rover is used together with our fast summation algorithm and adaptive isocontouring to efficiently visualize density contours of proteins in a multiresolution fashion.

All the above algorithms are implemented in a public domain software package called TexMol.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

Proteins, together with sugars, fats, oils, RNA and DNA are molecules which form the structural and functional building blocks in a cell. The first protein, the myoglobin of sperm whales, was crystallized by John Cowdery Kendrew and Max Ferdinand Perutz in Cambridge in the 1950s. Better imaging methods, including higher resolution x-ray diffraction, nuclear magnetic resonance and cryo-electron microscopy has yielded nearly 40,000 atomic structures. [1] A classification of proteins is given by Dr. David Goodsell in [67], based on their activity as molecular machines, performing complex tasks to enable life processes. These include functions performed outside the cell, by familiar molecules such as insulin, glucagon, antibodies and viruses, cell membranes composed of lipids, transport molecules like hemoglobin, enzymes that act as catalysts to regulate chemical reactions, DNA that store genetic information, protein factories composed primarily of ribosomes and other RNA, and structure proteins including microtubules and proteins like myosin which move along actin filaments.

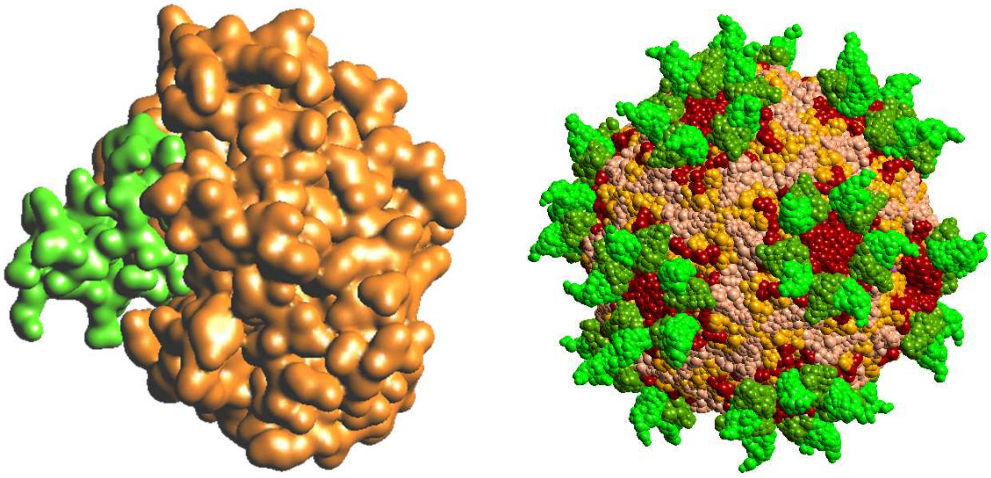
[1] The RCSB Protein Data Bank [21] is a database containing these structural descriptions.

Definition 1.1.1 (Protein)**.** Proteins are stable, folded collections of one or more polymers of amino acids and form the main building blocks in our cells. These organic compounds consists mainly of carbon, nitrogen, oxygen, hydrogen and sulphur atoms.

Definition 1.1.2 (Residue)**.** Amino acids ($NH_2C^{\alpha}HRC'OOH$) consist of an amino group (NH_2), a carboxyl group ($C'OOH$) and one of 20 side chains known as residues (R) that attach to the central carbon atom C^{α}.

The function of such proteins is expressed through their mutual structural interactions. Inhibitors bind to enzymes to reduce their rate of reaction. Immunoglobulins attach to antigens like viruses, to signal that it is a foreign object in cells. See figure 1.1 for two such examples: trypsin with its inhibitor and human rhinovirus 1RVF.pdb [2] with an immunoglobulin. Large and small ribosomal subunits combine and 'move' along mRNA strands during the formation of new proteins. Hence study of protein-protein interaction through computational modeling and visualization is an important key in understanding life processes. Efforts in structural proteomics have lead to a rapid increase in the number of three-dimensional structures of individual proteins. Moreover, knowledge of networks of interactions and signaling pathways is also expanding rapidly through genomic and proteomics approaches. Still, our picture of the structures of both stable and transient protein interactions

[2] Atomic structures in the PDB data base are usually denoted 4 characters and a .pdb extension.



(a) Trypsin and a inhibitor

(b) Human rhinovirus and immunoglobulin

Figure 1.1: In the first image, we show trypsin (brown) docked with an inhibitor(green). The second image shows the human rhinovirus 1RVF.pdb docked with an immunoglobulin. The virus' 4 chains are in shades of red to yellow, while the immunoglobulin two chains are in shades of green. (The two images are not to relative scale, the virus being much larger.)

lags behind. Efforts in crystallizing macromolecular complexes have met with limited success, and hybrid experimental approaches, utilizing cryo-electron microscopy and crystallography or NMR to give structural details of complex assemblies are evolving. However, along with these experimental methods, there is a growing need for efficient and robust computational approaches to predicting the structures of protein interactions. These approaches, known as protein-protein docking, have been developing over the past decade.

1.2 Protein Docking

An important step towards understanding protein-protein interactions is to computationally model and predict if and where two given proteins can bind. Specifically, we are interested in **protein-protein docking**, which can be defined as computationally finding (if it exists), the best relative transformation and conformation of two proteins which results in a stable complex, reproducible in nature. The two main aspects of docking hence include search algorithms and scoring potential complexes.

Affinity functions and scoring: Shape complementarity along the interface is seen to be important for docking, leading to the idea of a 'lock-and-key' fit between the two proteins [99]. Other factors which contribute to the formation of a complex include electrostatics, hydrophobicity, hydrogen bonds and solvation energy. Electrostatics plays a role in long range interaction due to partially charged protein and solvent atoms. The change in energy due to displacing water molecules from the interface is known as desolvation energy. These functions, together with shape complementarity are known as affinity functions. The docking problem can be considered as a search for stable minimum energy complexes.

Search algorithms: Given the affinity functions, and a scoring method, a search needs to be made over all of transformation and conformation spaces to find where the two given proteins best fit.

Rigid, soft and flexible docking: Proteins are known to be flexible in nature, both bending along their backbone chains and residues. (See figure 4.3 for different torsional angles through which a protein can typically flex.) This dynamic nature leads to a combinatorial increase in search space for the docking problem. Docking procedures, considering proteins as rigid is used as a first step in the complete docking pipeline, and used to reduce search space to potential docking sites [32]. The problem of docking, where both proteins are considered to be rigid bodies is known to be *rigid docking*. Algorithms which allow for slight interpenetrations while computing the docking score mimic flexible side chains and small backbone movements. In grid based methods, it is common to smooth out the boundary of molecules to implicitly account for its flexibility and conformational changes during binding. This is commonly referred to as *soft docking*. There have been few attempts at docking proteins while allowing for their flexibility recently, leading to *flexible docking* procedures.

Protein-protein docking and protein-ligand docking: When studying protein interactions and molecular assemblies, large proteins (generally greater than 1000 atoms) are considered. For example, the large and small ribosomal subunits contain more than 50,000 atoms). Due to stable folded regions, such molecules are considered to be fairly rigid or modeled with domain motions. They have side chain movements which are important to consider while docking. On the other hand, protein-ligand docking is more concerned with docking

proteins with small molecules, called ligands, typically in the order of tens of atoms. Such ligands, being small and in general more unrestrained, exhibit a high degree of backbone flexibility. See [72] for a more comprehensive analysis.

Docking guideline Due to the large search space, docking is traditionally performed in stages before a minimal set (of say 10) possible stable complexes are presented. In the most basic stage, rigid docking is performed, usually with shape complementarity and a simple model of electrostatics as the affinity functions and a large set of possible conformations are recorded. The next stage would be to further limit the search space by introducing flexibility and more precise affinity functions. Molecular dynamics is then performed to examine which possible conformers are stable and can possibly form a complex in nature [3]. In this thesis, we are primarily concerned with the first stage of this pipeline, including protein-protein docking.

1.2.1 Importance and Applications

Computational modeling and visualization of protein interactions has important applications in both industry and research.

- *Protein synthesis:* Creating more efficient molecules like hemoglobin, requires us to model the protein and increase its specificity and activity with other proteins.

[3] As discussed in the 3rd Conference on Modeling of Protein Interactions in Genomes, 2005, Lawrence, KS, USA

- *Analysis of molecular assemblies:* Biomolecular interactions can be modeled to understand the behavior, function and activity of molecular assemblies. Docking has been used to study the motion of ribosomes during protein synthesis. Structure of large compounds like viruses and microtubules can be deduced and studied given just one unit.

- *Computational virtual screening for drug determination:* By better understanding protein-protein docking, and then extending the search algorithm to drug like molecules, drug design [137] can be vastly improved from trying random candidates in the lab to computationally docking a large database of potentially active ligands to a receptor protein. This also helps in improving the screening process for new drugs.

- *Expensive to create and test in laboratories:* Of the approximately 40,000 atomic structures present in the PDB, only a few are complexes. It is not feasible to synthesize and extract atomic structure information of possible complexes, especially when virtual screening involves nearly millions of ligands. Synthetic protein design again involves studying its interactions with a multitude of other proteins, and involves testing a variety of residue changes.

1.2.2 Computational Challenges

Here we list only challenges that arise in computing and visualizing these interactions, and apart from the difficulty of improving models for affinity functions.

- *Size of the molecules:* As the size of imaging datasets grow, we are faced with new challenges to computationally model and visualize the structure and interactions of proteins. Viruses and microtubules typically contain millions of atoms. Algorithms which compute their surfaces, and determine their interactions with other proteins and visualize them needs to be scalable and parallelizable.

  - Algorithms to compute molecular surfaces and density functions do not scale, or are not modeled in a hierarchical manner to handle large molecules.
  - The search space for docking is very large. Most algorithms compute over grids which are typically 128^3 in size for small molecules and cannot handle large molecules, either in memory cost or computational complexity.
  - Interactive visualization of interactions of these large structures have previously not been done due to limitations of traditional graphics algorithms.

- *Flexibility:* Rigid docking algorithms typically search over the entire space with a fine discretization and can be performed in under a week on most workstations. Flexible molecule docking, even with a reasonable number of degrees of freedom, immediately prohibits a full blown search.

- *Time scales:* Docking and, in general, protein interactions occur in millisecond to microsecond time scales. Molecular dynamics simulations can

only cope with up to nanoseconds and are quite inadequate in performing docking.

1.3 Contributions

Below we list some of the contributions of this thesis in achieving our goal of modeling and visualizing protein-protein interactions. In later chapters, we will explain each in detail.

- **Model for handling protein-protein interactions** In this thesis, we provide a mathematical model for soft docking [29], provide data structures to represent affinity functions and surfaces, develop algorithms to compute the docking score with error bounds and novel techniques to visualize, with good per pixel shading, the protein surfaces, functions and interactions.

- **Molecular structure representation**

  - *Grid based molecular surface*

    Computing and representing molecular interfaces has traditionally required complex geometrical data structures like alpha shapes where as adaptive and uniform trilinear grids are commonly used in various simulations involving interactions of molecules or computation of electrostatics and other energy terms. We provide a signed distance function based algorithm using such adaptive grids to effi-

ciently compute molecular surfaces and properties like area, volume, curvatures, surface atoms and other surfaces. For M atoms (including B boundary atoms), smallest grid spacing h, grid length N, VDW radius r and solvent radius r_p, the timing complexity is $O(N^3 + O(log(N^3)B)) + O(M(\frac{2(r+r_p)}{h})^6 C)$, C is cost(dist(patch, voxel)).

– *Implicit molecular function representation and computation*

We present a new data structure called the Flexible Chain Complex (FCC) which describes the molecules flexibility, electron density and charge distribution. The volumetric functions of electron density, electrostatics and hydrophobicity are represented as summations of atomic kernels. For a molecule with M atoms, where the Fourier coefficients have a decay of the type $1/\omega^3$, we present an $O(M + n^3 \log n + N)$ time, Fourier based algorithm to compute N approximate, irregular samples of a level set surface and its derivatives within a relative L_2 error norm ϵ, where n is $O(M^{1/3}\epsilon^{1/3})$. Specifically, a truncated Gaussian of the form e^{-bx^2} has the above decay, and n grows as $\sqrt{b}$. In the case when the N output points are samples on a uniform grid, the back transform can be done exactly using a Fast Fourier transform algorithm, giving us an algorithm with $O(M + n^3 \log n + N \log N)$ time complexity, where n is now approximately half its previously estimated value.

We also provide a fast update to computing the summation func-

tions when a part of the molecule moves. Let the movement of a domain d with c centers affect N_d output points. The total cost to update the summation of kernels function f when the domain d moves is $O((2m+1)^3 c) + O(\alpha^3 M log(\alpha^3 M)) + O((2m+1)^3 N_d)$, where $\alpha \approx 2, m \approx 3$ [17].

- **Docking**
  - *Soft protein-protein docking algorithm.* Given 2 proteins with M_1, M_2 atoms respectively, we present an $O(max(M_1, M_2) + n^3 \log n + \rho n^3)$ algorithm to find the top ρ peaks in the docking profile. We also show that for a summation of Gaussians model for the molecule where atoms are represented as Gaussians, n^3 is $O(max(M_1, M_2))$.
  - *Flexible protein-protein docking scheme.* A new data structure with a simple file format is provided to users to represent the flexibility in a protein in a hierarchical manner. This graph is also computed using a new algorithm based on results of Normal Mode Analysis. Using this data structure, we provide a multi-stage docking algorithm which effectively samples orientation and conformation space, while using a multiresolution representation of molecular structure. Given a potential interface, we provide a greedy heuristic algorithm to improve shape complementarity at the docked surface.
- **Visualization**

– *Imposter based visualization:* We present a new technique to render well defined curved surfaces like spheres, cylinders and helices using single quads on programmable graphics cards [13, 151]. This allows us to interactively render molecules with millions of atoms with per pixel shading, which has never been done before.

– *Miscellaneous rendering techniques*

  * Dynamic multiresolution molecular surfaces.
  * Adaptive mesh refinement data structure [133].
  * Time-varying volume visualization [157, 158].
  * Hierarchical basis compression [12].
  * Compressed data based algorithms [11].

1.4 Overview

There have been many approaches to perform protein-protein docking, both rigid and flexible. We will review some of the algorithms and characteristics of various algorithms in chapter 2. Since ours is based upon Fourier expansions, and Fourier Transforms Grid based algorithms are commonly used in the docking community, we will explore such grid based methods in greater detail in 2.1.4 and provide a complexity analysis.

We introduce protein structure and flexibility in 3. Our data structures used to dock and visualize are also introduced here. Shape complementarity/matching is an important requirement for docking/matching in practise.

This dual form (implicit and parametric) representation of shape is used for computing docking and visualizing the proteins. The parametric form is used to compute molecular skins and surfaces for docking, while the implicit form is used to formulate the docking problem in later chapters.

Chapter 4 contains a complete description of our docking algorithms. We will present the different affinity functions we consider, an error bounded algorithm for soft protein-protein docking, adaptive global conformation sampling of flexible proteins, and techniques to improve predicted docking interfaces.

We provide extensive results of our docking algorithms and detailed analysis of three sample cases taken through the pipeline in chapter 5 on results.

Visualization of these protein interactions, their interfaces, and properties provide both visual cues of the algorithms characteristics and knowledge of their shape and behavior. In chapter 6, we provide novel algorithms to render large molecules, including million atom viruses, using a mix of 2D and 3D texture maps as *imposters*, at interactive speeds and high quality.

A summary of our results and current, future work is discussed in the last section. For completeness, we provide some of the error bound derivations, expansions and discussion of our software in appendices.

Chapter 2

Related Work

Computational protein-protein docking has been studied from the late 70s, and progressed as both the number of known protein structures grew and computers became faster and more commonplace. A review of previous research in this area is provided below to identify the computational techniques used, the affinity functions developed, the success and failures of the algorithms over different data sets and to examine the bottleneck in each case. In particular, we mainly focus on and classify the algorithms used to perform structure based docking. Due to the complexities involved in docking flexible proteins, a slew of rigid docking algorithms were developed to be used as a first step in the docking pipeline. If we were to check all possible docking positions for two rigid proteins, we would be performing a continuous search over a 6D search space. Traditionally this has been considered as three degrees of rotation and three degrees of translation. Approaches to tackle this complexity include hashing techniques, surface feature mapping, Fast Fourier methods and Harmonics based approaches. Apart from such computational methods, knowledge of active sites and knowledge of the surface chemistry is also used to reduce the search space. Since our method follows from other grid based approaches, we will review them separately in §2.1.4.

2.1 Rigid Docking Approaches

Both grid based 6D searches and feature based searches have been used to perform rigid protein-protein docking.

2.1.1 Critical Point and Feature Matching Algorithms

One of the first docking algorithms, by Kuntz et. al. considered cavities on the surface of the receptor as potential docking sites [99]. They add spheres to cavities on the receptor and match them with spheres on the ligand. To handle the combinatorial complexity, they match pairs of spheres from each and consider all sets of 4 pairs that are structurally compatible with one another. All such sets are further investigated as potential docking sites (myoglobin - heme and thyroxine - prealbumin were examined). Given n spheres on the receptor, the time complexity of the algorithm is $O(n^4)$. A similar algorithm is shown by Connolly in [38] where he first computes *knobs,holes* in the surfaces of the proteins and matches them using a similar algorithm as above and various geometric heuristics. It was tested on the alpha, beta subunits in a hemoglobin tetramer and trypsin with its inhibitor, with partial success. The same set of examples were tested by Wang [168] with a slight modification of the algorithm. Initially, one pair of knob and hole is taken and the proteins moved such that geometrically oppositely oriented. Then the remaining degree of rotational freedom along that axis is sampled and tested. They used a grid based skin region, which has been further exploited in many other schemes.

Another approach to compute docking, using a graph theoretic ap-

proach is shown in [98]. Each atom of the receptor is paired with an atom of the ligand, and the distance between them is computed. Next consider an edge between two such pairs if the distances are close. This implies that the 4 atoms are structurally suited to fit together. Computing a clique of such a graph or simply its subgraph gives us potential docking sites, based on shape complementarity. Their algorithms worst case was reported to be $O((nm)^4)$ for n, m atoms in the system, but they also claimed a average behavior of $O((nm)^{2.8})$. Bipartite graphs were considered in [152] to match edges as above. A set of 100 complexes were docked using this approach and was found to be better than the bipartite graph matching used (then) in program DOCK [53].

2.1.2 Geometric Hashing

Pattern recognition is an important area of research in the image processing community. One of the main algorithms used there is determining *footprints*, and using them for search, leading to *geometric hashing* and the idea has been incorporated into protein docking. Lamdan and Wolfson in [100] introduced this idea to model based object recognition from 2D images. Matching pairs of atoms on ligands with those on the receptors and pruning searches based on interpenetration was implemented in [59]. For protein-protein docking, they felt the knobs and holes was better suited and again used geometric hashing to compute possible docking sites. For the ligand protein dock, they used three sets: Heme-myoglobin, NADPH-dihydropholate reductase and tyrosinyl adenylate-tyrosyl tRNA synthetase. For protein-protein docking they

used trypsin-its inhibitor and HIV-1 protease's subunits. They obtained the correct docking transformation in the top 250 and the top respectively. Their algorithm is also provided in [58, 130]. Normals were used in [131] to sharply improve their docking results for a set of 16 complexes. A comparison of various techniques and a parallel implementation is discussed in [108]. In [47], more results are presented using biochemical filters to restrict search to active sites.

2.1.3 Surface Geometry Matching

Direct matching of geometric surfaces has also been considered, instead of just feature points. Walls and Sternberg slice up the protein surfaces into patches and match them geometrically. Their algorithm, in [167], is more suited for spherical proteins where the mapping from surface to plane is easier. Given n_1, n_2 patches for each protein, and ρ, ω being the translational discretization in bringing them together and the discretization of the remaining degree of freedom, their cost is $O(n_1 n_2 \omega \rho)$ (They also perform additional matching at each step by moving one surface patch along the other). Bacon and Moult [8] perform docking of ligand surface patches to a protein active site using pattern matching over webs. Webs are concentric bspline rings with near uniform spacing, constructed over the molecular surface. Patterns are taken from the webs and least squares fitting is done to match the ligand surface to the active site. This is repeated for all ligand sampling, and done in a hierarchical two step process. Nine different sets, including bound com-

plexes, unbound and predicted structures were used in evaluation. Matching surfaces projected to cylinders was proposed by Helmer-Citterich and Tramontano. Each orientation was considered separately. For a given orientation, the pair is projected on to cylinders and sliced. In each slice, they produce points at equal orientations. The difference between neighboring points make up a vector. This stack of vectors creates a matrix, which is compared to the similar matrix for the other protein. To avoid bad sampling at the poles for the stationary protein, they use two different projections for it.

2.1.4 Grid, Fourier and Harmonics Expansion Based Algorithms

Apart from feature and surface matching algorithms, volumetric soft docking has become popular, especially to handle slight conformational changes. *Soft Docking* was introduced in in [84] as a full 6D grid-based search technique. Each orientation is considered separately. (They also provide a method to *uniformly* sample 3D rotational space). A surface point and its normal is replaced by a grid voxel. Thus the voxels from the two proteins are then overlapped for all translations, and a docking score, considering the number of overlaps, number of surface points in each voxel and their normals, is computed. Both bound docking (ternary complex of dihydrofolate reductase, NADPH, methotrexate, and trypsin and its inhibitor) and unbound docking (trypsin, its inhibitor and lysosyme and an antibody fragment) was performed and good results found in top 500 of their searches. Given a volume discretization of n^3 with m^3 rotations, this costs $O(n^6 m^3)$.

An obvious improvement in computational cost was introduced by Katchalski-Katzir, Shariv, Eisenstein, Friesem, Aflato and Vasker by using Fast Fourier Transforms (FFTs) in [91]. This led to a slew of papers along the same lines. In this technique, the proteins are embedded into grids. Grid points outside are assigned 0, on the surface 1, and inside, a negative and positive value for the different proteins. An overlap between the two maps, for a given orientation provides a good indication of shape complementarity. This overlap computation, over all translations is sped up using the convolution theorem, reducing the cost to $O(n^3 \log n^3)$. They tested their algorithm, successfully on 5 known complexes and with partial success on two native sets. Since this technique involves no speedup over the rotational space, in [120], the authors show how hydrogen bonding requirements can eliminate many orientations and incorrect matches. They give good results on a large number of complexes.

While the previous method included shape complementarity in the correlation based search, a model to perform electrostatics matching was shown by Gabb, Jackson and Sternberg in [60]. We use the same electrostatics model in our docking's affinity functions and will introduce it later in §4. Their results showed that addition of electrostatics improves their search on 10 different complexes. A post filter based on finer searches was also employed. Pair potentials, statistical scores to contact residues, used in protein folding was then used to improve their docking scores of the same set of complexes [126]. A parallel program DOT [116] uses a similar model of shape and electrostatics

potentials, but computes the potential using a linearized Poisson-Boltzmann solver. Chen and Weng introduced the program ZDOCK, which utilizes a third affinity function of desolvation free energy. They use Atomic Contact Energy, computed by examining existing crystal structures and use it to measure the free energy change in replacing a protein atom - water contact with an protein atom-protein atom contact. They tested their algorithm on 24 systems, obtained the correct structure for three and others in the top 2000. More details on their program and results are provided in [32].

From Wrigger's Lab, there have been three different approaches to matching protein structures and imaging data, all of which can be applied to protein-protein docking. In [174], they used the conventional Fast Fourier transform based convolution to speed up the three degrees of translation, and called their algorithm Fast Translational Matching. Fast Rotational matching shows how to speed up the three degrees of rotation instead, using spherical harmonics expansions in [94]. All of the above have a time complexity of at least $O(n^3 \log n^3)$ and require a large grid, with $O(n^3)$ memory. In [93], they introduced a novel way of speeding up five degrees of freedom using similar expansions and the Fast Fourier Transform. Again, using spherical harmonic expansions, the correlation function is expressed in terms of five degrees of rotation and one degree of freedom as

$$\sum_{ll'mnm'hh'} (-1)^n d^l_{nh} d^l_{hm} d^{l'}_{-nh'} d^{l'}_{h'm'} e^{i(n\sigma+h\eta+m\omega+h'\eta'+m'\omega')} I^{ll'}_{mnm'}(\rho)$$

where $\sigma, \eta, \omega, \eta', \omega'$ are rotational degrees of freedom and ρ is the translation between the proteins. d is related to the Legendre functions. This is done

for all intermolecular distances and across the volume function. This results in a computational cost of $O(n^9)$ and a memory cost of $O(n^6)$, assuming all discretizations are the same (However this need not be true). Ritchie and kemp [144],[143] expand their scoring function in terms of real spherical harmonics, leading to similarly high computational costs as shown in [29].

2.1.5 Rotational Space Sampling

In most of the previous algorithms, there is a need for sampling rotational space. It is a well known problem and optimum sampling is still an area of research as an exact solution does not exist. In fact, the common Euler angles sampling is highly defective as it samples too finely at the poles. The following three papers describe simple techniques to obtain rotational sampling: 1. Lattman in [101] used euler angle combinations. Consider 2 rotations R_1, R_2 with euler angles a_1, a_2, a_3 and $a_1 + d_1, a_2 + d_2, a_3 + d_3$. Let there be a quaternion which rotates through an angle q to take R_1 to R_2. Then the angle q is expressed in terms of a_i, d_i. They show that if instead of Euler angles (e_1, e_2, e_3), if we use $(e_1 - e_3, e_2, e_1 + e_3)$, then the angle q can be used to better sample the angles locally in an equispaced manner. Kuffner in [97] used random sampling of both euler angles and quaternions. In euler angles, one angle is sampled uniformly from $-\pi$ to π, another from $-\pi/2$ to $\pi/2$. Now if we sample the third from $-\pi$ to π, we will be sampling more dense at the poles. So instead, they take random samples of the arccos function, which samples less at the poles and more elsewhere. Yershova and LaValle in [177]

sample regular polytopes surfaces and use it to sample spheres.

2.1.6 Other Methods

A simulated annealing method, by choosing angles in discrete 45 degree steps and translations of 2A is used in [179] to perform a random walk and dock proteins. In [34], residues are approximated as spheres and the docking broken down as finding 5 rotations and a translation. The rotational space is sampled using simulated annealing. TreeDock by Fahmy and Wagner goes through all pair of atoms in both proteins and tries to align through the remaining degrees of freedom, using multidimensional search trees. They report excellent results when provided with an active sites or possible set of sites.

2.2 Flexible Docking Approaches

Many proteins are known to be flexible (see [35] for an example of the HIV-1 protease flexibility simulation). Small molecules, especially drug like compounds, exhibit a greater flexibility, especially along the backbones. Protein-protein docking on the other hand is more concerned with side chain movements, and is accounted for (to some extent) in *soft docking*. Here we briefly survey flexible docking algorithms. For unbound docking (where each molecule has been crystallized separately and probably exist in different conformations that compared to their complexed state), better flexibility models

will help in improving the docking score [1]. There have been only a few methods to handle flexibility in protein-protein docking. The more computationally feasible problem of protein-small molecule (or commonly ligand) docking has been tackled often, especially for virtual screening. Below, we will describe both protein-protein and protein-ligand algorithms, since they are mathematically equivalent. We will especially note when a method has been used for protein-protein docking.

2.2.1 Global Search Methods

Global search strategies have been based on energy minimizations, heuristics based search methods, and geometric identifications of cavities indicating possible active sites. In DOCK [99] and later [44], receptor binding sites were identified as cavities and the complementary space represented as spheres. Fragments of the ligand were separately bound to the active site using various distance heuristics between atoms and spheres. Fragments were then incrementally selected to form the entire ligand. An incremental approach based on shape [104], [171](HammerHead) and properties of the molecules FLEXX [140] is used to dock fragments, pruning the exponential search by retaining only a fixed set of possible conformers at each step. Other global search techniques include hydrogen bond pattern based search [124], genetic algorithms [86](GOLD), [85, 88, 132], monte-carlo/simulated annealing

[1] according to the Abagyan Lab, TSRI, 'Only about one third of the protein complexes can be docked without serious considerations for the induced conformational changes upon docking.'

[68](AUTODOCK),[26],[6], molecular dynamics [129] and evolutionary programming [63]. Genetic algorithms have been successfully used in protein-ligand docking, where it is important to consider the flexibility of the ligand. Gold, which is a collaborative project between Sheffield University, Glaxo-Wellcome and the Cambridge Crystallographic Data Center (CCDC) is one such program [85, 86]. Its scoring function has four components, hydrogen bonds, van der Waals energy, ligand internal Van der Waals energy, and ligand torsional strain [39]. Quick Explore (QXP) from Novartis Pharmaceuticals uses monte carlo energy minimization algorithms (MCdock [119] and Fulldock [96]). FlexX is a software that allows incremental construction of the docked complex [139, 140]. FlexE allows docking calculations to be performed using an ensemble of protein structures. FlexX was developed by Markus Lilienthal at BioSolveIT GmbH and Prof. Dr. Matthias Rarey at the Center for Bioinformatics (ZBH) of the University of Hamburg. AutoDock [68] from TSRI is another genetic based algorithm which uses both Simulated Annealing and a newer Lamarckian Genetic Algorithm. See [127] for the performance of different algorithms in AUTODOCK. Steered molecular dynamics, using a visualization and feedback toolkit has also been studied in SMD [107].

2.2.2 Backbone and Domain Movements

Hinge bending in either protein or ligand is also used in docking in [145], accounting for domain movements. Their algorithm is based on geometric hashing of triplets of atoms in ligands and receptors, using Kuntz model for

protein, ligand description, allowed for induced hinge rotations, and showed results for large protein-protein docking also. Conformations are sampled using a coarse set of values for torsion angles of rotating bonds in [169]. Those conformations which do not form severe steric overlaps are used in a rigid body docking. Torsion and bond angles are sampled and matched using the α-shapes of the molecules [9].

2.2.3 Side Chain Flexibility

Flexible side chains are more commonly modeled in protein-protein docking than movements in the backbone. Using rotamer libraries, and a greedy heuristic or branch and cut algorithm, [4] performs docking of proteins with flexible side chains as a second step to rigid protein-protein docking. Similar discrete side chain conformations were searched using a dead end elimination approach and A^* trees in [102]. By classifying residues as active and inactive residues, and clustering them into connected graphs of interacting using their rotamer libraries, SCWRL is able to assign low energy conformation rotamers efficiently [28]. A recent algorithm TreePack by Xu and Berger claim to run up to 90 times faster than SCWRL3 [176]. Analysis of the movements in side chains as a function of their size and flexibility was studied in [180], where they conclude that side chain motions are generally small to avoid steric clashes. A combination of the pseudo-brownian Monte Carlo minimization followed by flexible side chain docking using ICM was tested on a variety of bound and unbound complexes in [56]. Apart from backbone and side chain

movements, loop flexibility at known active sites is handled using a Monte carlo, simulated annealing based docking approach in [20]. The *connexions project* at http://cnx.rice.edu/content/m11464/latest/ maintains a summary of flexible docking algorithms.

2.3 Summary

CAPRI (Critical Assessment of PRedicted Interactions) is a blind test of protein-protein docking algorithms [82]. Initiated by Kim Henrick, Joel Janin, John Moult, Lynn Ten Eyck, Michael Sternberg, Sandor Vajda, and Shoshana Wodak, it has provided a means for different groups of researchers to test their docking algorithms on yet unpublished complexes, given the two individual proteins. They emphasize the need for better scoring functions and techniques to handle conformational changes [125]. Many of the successful algorithms used in the docking predictions had an initial rigid docking stage, performed using the Fast Fourier Transform or Spherical Harmonics approach. We will provide a new approach to compute the convolution search function accurately, efficiently and with formal error bounds.

2.3.1 Importance of Molecular Surfaces

In all the docking approaches listed above, shape complementarity was the main affinity function used to predict docking sites. The geometric fit between proteins is used as a primary filter, before better scoring functions are introduced to score potential complexes. Initial approaches defined the sur-

face using spheres. Then the binding site and ligand was represented as sets of spheres [99]. Connolly's algorithm [37], [36] is to compute the molecular surface was used in many docking approaches. Contact surfaces between domains of methemoglobin was studied in [71] and proposed as a affinity function to use in docking. Soft docking [84] uses a smooth region about the surface to allow for minor conformational changes. Ritchie develops his docking algorithm by introducing a spherical harmonics based parametric surface definition [142]. Hence, as a first step to protein-protein docking, we first introduce new algorithms and data structures to represent molecular surfaces, structures and functions in the next chapter.

Chapter 3

Molecule Shape Representation

Molecule shape representation for protein-protein docking must satisfy the following requirements. Shape complementarity is the primary affinity function used in docking. Hence, the molecular surface representation needs to accurately and efficiently model the interface between proteins and between a protein and solvent. Other affinity functions include, but not limited to: electrostatics, desolvation and hydrogen bonds. Therefore, it should handle not only structure, but also functions defined in the volume and on the surface. Molecular representation, the search algorithm and scoring functions are three important parts of the docking framework, and are interlinked, requiring the representation to be amenable to the search algorithm. Fourth, to handle flexibility, fast updates of local changes should be possible. In this chapter, we introduce a new data structure called the Flexible Chain Complex (FCC), which supports a dual form representation for proteins and provide algorithms to compute these two surfaces.

3.1 Molecular Surface Definitions

Explicit surface definitions as the interface between the solvent and proteins have been given since 1970s. Since it is easier to handle implicitly defined models mathematically, different implicit approximations to these surfaces have been developed.

3.1.1 van der Waals, Lee-Richards and Richards Surface Definitions

The most common model for molecules is as a collection of atoms represented by spheres, with radii equal to their van der Waals radii. The surface of the set of spheres is known as the van der Waals surface. Lee and Richards introduced the concept of accessibility to the solvent. Proteins are not isolated, but commonly present in solutions, especially water. Also, the van der Waals surface contained too many internal atoms and patches which are not accessible by the solvent. Hence, Lee and Richards gave a new definition for the molecular surface as the surface accessible to the solvent [105]. They modeled water molecules as spheres with radius 1.4Å, and considered the locus of the center of one such 'probe', as it rolled along the protein surface as the Solvent Accessible Surface (SAS). Richards then gave a more commonly used definition for molecular surface as a set of contact and reentrant patches in [141]. A probe solvent sphere, rolling over the atoms of a protein defines a region in which none of its points pass through. The boundary of this volume is continuous and defines a new molecular surface. This surface is composed

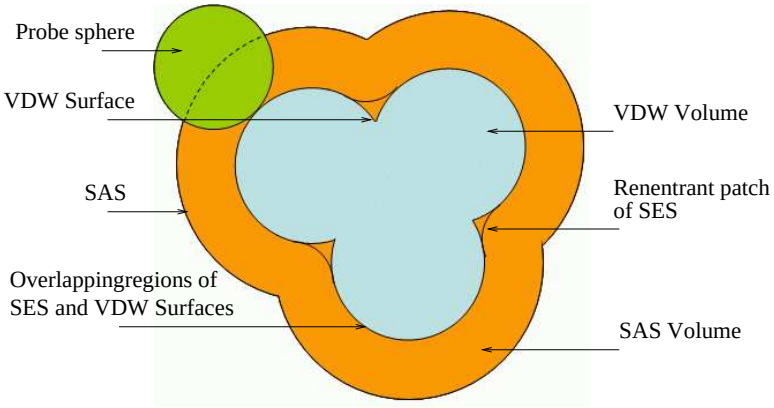


Figure 3.1: The different molecular surfaces and regions are shown for a 3 atom model in 2D. The SAS surface is the locus of the center of the rolling probe sphere. The VDW surface is the exposed union of spheres representing atoms with their van der Waals radii and contains the VDW volume. The lower side of the rolling probe defines the smooth SES which contains parts of the VDW surface and reentrant patches. We also define the SAS volume as the region between the SAS and SES. The region between the SAS and VDW volumes is later refered to as the SES volume.

of convex patches where the probe touches the atom surfaces, concave spherical patches when the probe touches more than 2 atoms simultaneously and toroidal patches when the probe rolls between two atoms. Connolly called this as an alternative definition of the SAS surface in [37], but is now commonly known as the Solvent Contact Surface (SCS), or Solvent Excluded Surface (SES) or simply the Molecular Surface. These surfaces, for a 3 atom example is shown as a 2D cross section in figure 3.1. We also provide analogous volume definitions for each surface.

3.1.2 Implicit Models of Protein Structure

The above explicit surface definitions suffer from discontinuities and are not easy to incorporate into computational methods to obtain density and charge functions and led to volumetric function definitions to approximate the electron density of a protein. An appropriate isosurface of this function is often used as the molecular surface. These volumetric functions have been used to model and compute functions including electron density, electrostatics, solvation energy, forces and hydrophobicity. Gaussians have been used to model orbitals of electrons in quantum chemistry [24] and used in shape description [66]. Blinn introduced representing atoms as Gaussians for visualization purposes in [22]. Mezey, in his book Shape in Chemistry [123] (and a series of papers [121, 122, 166]), used Gaussians to model atomic electron density, and uses it for topological analysis of the surface patches. While he uses general anisotropic Gaussians, isotropic Gaussians were used by Grant and Pickup in [70] where they compared the volumes of the hard sphere model to those of their new Gaussian model. Dielectric values differ for the solvent and the proteins interior. To prevent sharp boundaries from introducing instability in the solvers, different smooth definitions are applied, leading to yet another different definition for the molecular interface. Im, Beglov and Roux use $0, 1$ in the interior and exterior of the protein for the volume exclusion function, but a smooth transition at the boundary: $-1/(4w^3)(r - R_\alpha + w)^3 + 3/(4w^2)(r - R_\alpha + w)^2$, in a window of size $2w$ for an atom α of radius R_α [78]. Im, Lee and Brooks use product of kernel functions in [79] to provide another smooth definition

for the volume exclusion function. Gabdoulline and Wade mention that exponentially decaying functions provide a better asymptotic behavior of electron density, but do not give any further reasons [61]. Spherical harmonics based expansion for surfaces of general shape molecules is described in [49], giving a surface parameterization. But for higher resolution representation, they also prefer Gaussian kernels ([50]). The variance and height is adjusted 'so that the sphere defined by the atoms van der Waal's radius contains 2 standard deviations of the density'.

Both the explicit SES model and the implicit sum of Gaussians function have their advantages in docking algorithms. Since shape complementarity is an important feature in protein-protein docking, it is essential to compute protein surfaces efficiently and accurately. On the other hand, the implicit sum of Gaussians function is a convenient model to use due to its desirable properties, including smoothness, convolution property and parameterization. Hence, we introduce a new dual form representation for molecular structures and properties.

3.1.3 FCC for Dual Form Representation

Proteins have a naturally occurring backbone, forming chains which flex through their torsion angles as shown in figure 4.3. Structural (shape) and functional properties are described as a labeled *sheath* around the central *nerve*. This combined labeled representation of a *nerve* and a *sheath* is used to model a flexible protein's structure and properties and is referred to as a

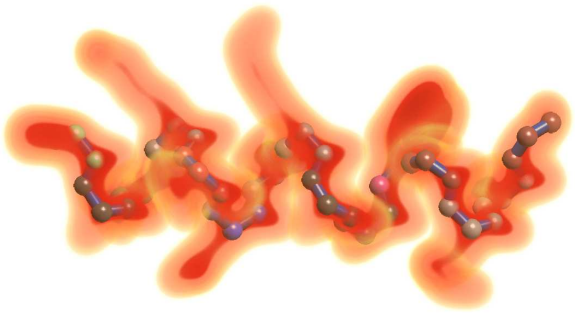
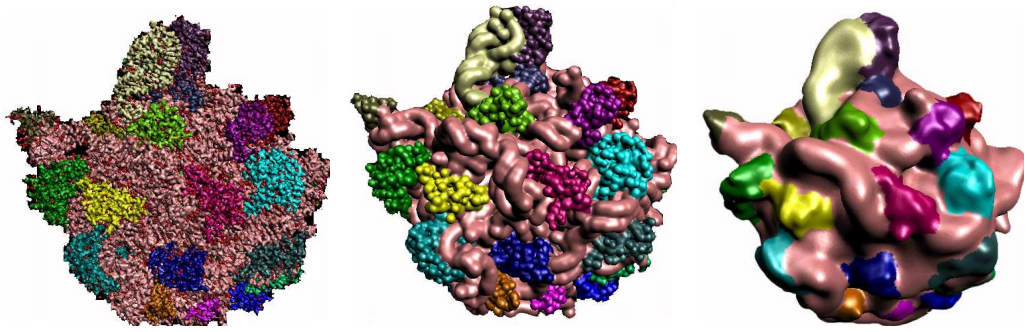


Figure 3.2: Flexible Chain Complex: Combined rendering of a part of a protein, showing the backbone (chain) together with the high density volumetric beads formed by the functional groups (residues) protruding outwards from the chain.

Flexible Chain Complex (FCC). A part of the backbone of a protein is shown with the surrounding electron density cloud in figure 3.2. The hanging residues along the chain form the *beads* of the chain, and the backbone rotates along its torsional angles, giving the chain its flexibility.

The chain complex consists of labeled vertices, edges and faces. Atom or pseudo atom positions form vertices in the compelx. Atom positions are obtained typically from the PDB files. For pseudo atoms, we use the centers of a set of enclosing spheres which represent the finer level using some error norm like the Hausdorff error for clustering. This is again from the PDB or from the hierarchical complex formed by clustering the finer resolutions to a directed acyclic graphs. The faces are the residues or clusters of residues. These elements are labeled with positions, lengths, and areas. Ranges for flexible angles, lengths are marked. The surrounding volume, sub volumes and surfaces of a biomolecule are used to represent shape, volumetric properties



(a) Atom level (b) Residue level clustering (c) Coarsest level

Figure 3.3: LOD volume rendering of a large ribosomal subunit (1JJ2.pdb). The first figure shows an atomic scale model. The spread of density around a pseudo atom representing a residue of the hierarchical chain complex is varied in the second and third figures.

(like electrostatics, hydrophobicity) and surface properties (like curvatures). Both the skeletal and the volumetric features are represented in a hierarchical fashion. We have a biochemical based static hierarchy of the molecules, with atoms at the finest resolution. Groups of atoms are collapsed to form residues and residues form secondary structures. Chains consist of a set of these secondary structures. Once a flexible chain complex hierarchy is rebuilt due to dynamic changes in the molecule, the implicitly defined volumetric and surface properties can be quickly updated. Explicit volumes can also be extracted in a hierarchical fashion. When we have a hierarchical representation of a FCC skeleton, we implicitly have a hierarchical representation of the surrounding differentiable sheath. In figure 3.3, we show the large ribosomal subunit at three different levels of a hierarchy. We base all our surface construction and docking algorithms on the FCC data structure.

3.2 Adaptive Grid Based Surface Construction

We provide a new algorithm to compute the molecules SES and other related properties, which we will use in the docking framework. Our main goals are efficient representation for operations required during docking, accurate surface definition and ease of implementation.

3.2.1 Related Work

Since Richards introduced the SES definition, a number of techniques have been devised to compute the surface, both static and dynamic, implicit and explicit. Connolly introduced two algorithms to compute the surface. First, a dot based numerical surface construction and second, an enumeration of the patches that make up the analytical surface (See [36], [37] and his PhD thesis). In [165], the authors describe a distance function grid for computing surfaces of varying probe radii. Our data structure contains approaches similar to their idea. A number of algorithms were presented using the intersection information given by voronoi diagrams and the alpha shapes introduced by Edelsbrunner [51], including parallel algorithms in [164] and a triangulation scheme in [1]. Fast computations of SES is described in [146] and [147], using Reduced sets, which contains points where the probe is in contact with three atoms, and faces and edges connecting such points. Non Uniform Rational BSplines (NURBs) descriptions for the patches of the molecular surfaces are given in [15], [14] and [16]. You and Bashford in [178] defined a grid based algorithm to compute a set of volume elements which make up the Solvent

Accessible Region.

3.2.2 Signed Distance Function based Family of Surfaces

We define a volume function Φ and use its contours to provide a family of molecular surfaces. Consider the union of atoms of the molecule $\cup B$. Inflate each atom b in this set by the probe radius (solvent radius) r_p to give the new complex $\cup B_{r_p}$. Let its boundary be Γ_B. Let Φ define the signed distance function of ΓB, such that the interior (closer to van der Waals) is given a positive sign. Let all regions within the atom (see [178] for definitions) be given a constant positive high value H.

Observations:

- Isosurfaces S_I with isovalues $I : 0 \leq I \leq H$ form a family of surfaces.
- $\Gamma_B = S_0$, as defined by Lee and Richards, is the SAS of the molecule.
- S_{r_p} is the SES.
- $S_{x \to H^-}$ is the van der Waals surface.
- $\{x : 0 \leq \Phi(x) \leq H\}$ defines a volume exclusion function, which can be convenient to use in electrostatic computations.
- The region $\{x : -r_p \leq \Phi(x) \leq r_p\}$ has a high probability for the presence of surface atoms of a protein docked to the current one.

The above observations point to the obvious advantages in using such a definition for our molecular structure representation for docking. Let us further examine some of them in detail.

$\mathbf{\Gamma_B = S_0}$ is the SAS, and $\mathbf{S_{r_p}}$ is the SES: By definition of the SAS, it is the locus of the center of the probe as it rolls over the spherical atoms of the protein. But it should be noted that the grid based definition also includes holes, which may be removed if necessary. The SES surface is always defined by points on the probe. It is in fact the boundary of the region accessible to any part of the probe radius. Hence, it is always at a constant distance of r_p away from the locus of the center. Therefore, our third observation follows. Again, holes are included in our definition and need to be removed if required.

$\{\mathbf{x} : \mathbf{0} \leq \mathbf{\Phi(x)} \leq \mathbf{H}\}$ provides a volume exclusion function: Volume exclusion functions are used in setting up dielectric constant for electrostatic computations. The twin requirements of smoothness at the boundary and accuracy in modeling the SES are not met by many of the definitions in practise today. Our definition is provides a 'sufficiently' smooth function around the SES (Φ is smooth in the radial direction), and contains the SES within it.

Isosurfaces $\mathbf{S_I}$ with isovalues $\mathbf{I : 0 \leq I \leq H}$ form a family of surfaces At the extremes isovalues, we have the SAS and the VDW surfaces, and the SES lies in between them at an isovalue of r_p. We show the SES surface and other surfaces surrounding it in figure 3.8.

Interface of docked ligand is in the region $\{\mathbf{x} : -\mathbf{r_p} \leq \mathbf{\Phi}(\mathbf{x}) \leq \mathbf{r_p}\}$: For good shape complementarity, as observed in docked complexes, atoms of the ligand must lie close to the surface of the protein. The above 'skin' definition provides a functional representation for such a region, as it defines the region where a probe sphere is in touch with the protein.

3.2.3 Algorithm

Let us consider a grid G in which the molecule is embedded to have a maximum and minimum grid spacing, h_{max}, h_{min}. Let the dimension with the lower resolution be N^3. Initially, the grid is uniformly divided using h_{max} as the grid spacing. Then:

- **Top down subdivision:**
    1. Insert each atom $b_i, i = 1..M$ into G, subdividing if necessary.
    2. With each insertion, update locally, points $\vec{p} \in G$ as belonging to V_{SAS}, V_{VDW}.
    3. Compute the boundaries S_{VDW}, S_{SAS}.
- **Bottom up collapse**
    1. New points created by the previous steps in the grid, and buried in atoms interiors are collapsed to make the grid sparser in a bottom up fashion.

- For each point $\vec{p}$ around a point classified as S_{VDW}, search in a local region with extent r_p for a S_{SAS} boundary cell. Find the closest distance of the point from the S_{SAS} boundary.

  Details on each step is given below.

Vertex classification Each atom is inserted into the grid. If we start with a single node in the adaptive mesh, subdivision is performed as we insert each atom in an adaptive manner. With the insertion of each atom, the vertices around the center of the atom are classified as belonging to inside the V_{SAS} or the V_{VDW}. Vertices classified as V_{VDW} are fixed while vertices marked V_{SAS} could be updated with the insertion of new atoms. We use the method described by [7] for sphere-cube intersection tests. The cost of this insertion is linear in the number of atoms and cubic in the resolution of the grid: $O(Mh_{max}^3)$.

Boundary cell detection We examine the classification of the eight corners of each cell of the grid. If some vertices belong to the inside of a volume and others to the outside, we mark the cell as a boundary cell. This operation is linear in the number of cells of the grid $O((N-1)^3)$. Each boundary cell is given an index.

Adaptive subdivision of S_{SAS} Each boundary cell which contains more than three atoms contributing to it is subdivided up to a user defined res-

olution. The index of each atom intersecting a cell is kept in a linked list associated with that cell. Using that list, we classify each subdivided vertex as belonging to the interior of the V_{SAS} or not. Using a technique similar to obtain boundary cells, we generate a list of finer boundary cells in the subdivided cells. The maximum cost of this operation is $O((N-1)^3(h_{max}/h_{min})^3)$, although the average case cost should be much smaller as only the boundary cells are involved.

S_{SES} computation The cells around each vertex in V_{SAS} and S_{VDW} is searched for the S_{SAS}. If there is a cell with only one intersecting atom, we find the exact distance from the vertex to the spherical patch of S_{SAS} in that cell and stored at the vertex. If a closer distance it is found, the stored distance is updated. If we are searching a cell which contained more than one atom, and hence subdivided, we just take the minimum distance from the center of all the subdivided cell to the vertex in question as the distance of the spherical patch in the cell to the vertex. The cost of this search will vary as r_p^3, the number of boundary van der Waals cells in the volume and the accuracy desired (as provided by h_{min}).

3.2.3.1 Spherical Patch Intersection

Let us define a sphere as having a center $\vec{c} = \{c_x, c_y, c_z\}$ and radius r. Define a cube with points $\vec{a}_1, .., \vec{a}_8$

Equation of arc of intersection of sphere and face of cube The intersection is always an arc of a circle. We will obtain the center, radius of the circle and the intersection points. We will consider only a face parallel to the xy plane. Other cases should follow similarly. The point of projection of $\vec{c}$ to the plane containing the face is $\vec{p} = \{c_x, c_y, \texttt{z coordinate of face}\}$. This point is the center of the circular arc. The radius using Pythagoras theorem is $\sqrt{r^2 - dist(\vec{p}, \vec{c})^2}$. The intersection points on the edges, if any is now the intersection of this circle with the line containing the edge, and checking whether the points lie within the edge.

Shortest distance of point to a circular arc Let the point be $\vec{p}$, the center, radius of the arc be $\vec{p_1}, \vec{r}$ and the two end points of the arc be $\vec{p_2}$ and $\vec{p_3}$.

Lemma The shortest distance of a point $\vec{q}$ in a plane to a circular arc in the plane is:

- Point is outside the infinite sector defined by the arc. The shortest distance is : $|\vec{r} - dist(\vec{q}, p_1)|$.
- Otherwise, the shortest distance $= min(dist(q, p_2), dist(q, p_3))$.

Let $\vec{q}$ be the projection of $\vec{p}$ to the plane containing the circle. Let d_1 be the $dist(\vec{p}, \vec{q})$ and d_2 be the shortest distance of the arc from $\vec{q}$. Thus, the shortest distance from $\vec{p}$ to the circular arc is $\sqrt{(d_1)^2 + (d_2)^2}$.

Shortest distance of point to a spherical patch Here the spherical patch is in a cube, bounded by circular arcs. Consider the circle a boundary arc is part of. The center of the sphere and this circle will form an infinite cone. Hence the collection of boundary arcs form a collection of infinite cones.

Lemma The shortest distance of a point $\vec{p}$ to a spherical patch in a cube is:

- Point is inside each of the infinite cones. The shortest distance is : $|\vec{r} - dist(\vec{p}, \vec{c})|$.
- Otherwise, the shortest distance is the minimum of the shortest distances of the point to each of the bounding arcs.

3.2.3.2 Complexity

For M atoms (including B boundary atoms), smallest grid spacing h, grid length N, VDW radius r and solvent radius r_p, the timing complexity is

- SDF initialization: $O(N^3)$
- Insertion of atoms: $O(M(\frac{2(r+r_p)}{h})^3)$
- Boundary atom detection:
  - Uniform grid traversal: $O(N^3)$
  - Sphere traversal: $O(M(\frac{2(r+r_p)}{h})^3)$
  - Octree traversal: $O(log(N^3)B) \leq O(log(N^3)M)$

- Patch voxel distance computation: $O(M(\frac{2(r+r_p)}{h})^6 C)$, C is cost(dist(patch, voxel))
- Isocontouring for visualization: $O(N^3)$

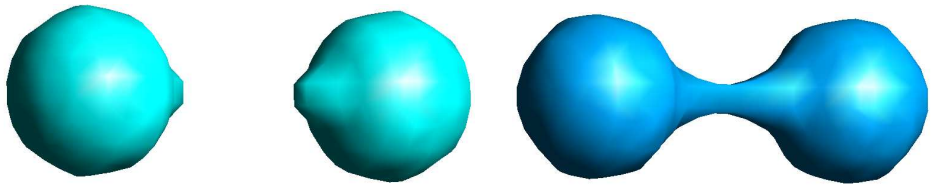
3.2.4 Self Intersections in Patch Complex Model

A patch complex (consisting of convex, concave and toroidal patches) can be derived using our adaptive grid structure and SAS sphere intersection enumeration. But the patch complex is known to have problems of bad intersections. According to lemma 3, 4, 5, 6 and 7 from Bajaj et al [18], there are only two possible self intersections that occur in the commonly used rolling ball model:

- A toroid can self intersect (Figure 10(a) in [18]).
- A concave patch can intersect with another in the case of a 3 atom model (Figure 9 in [18]).

In figure 3.4, we show the surface computed when two atoms are present, and moved close till they form a single surface patch. In the case of surfaces computed from the rolling ball model, we would instead get a self intersecting toroidal patch when the gap between the atoms becomes smaller than the diameter of the probe radius. This can be computed by looking at all pairs of intersecting SAS spheres, which is already given in our adaptive grids. To examine the intersection of two concave patches, we look at the three atoms

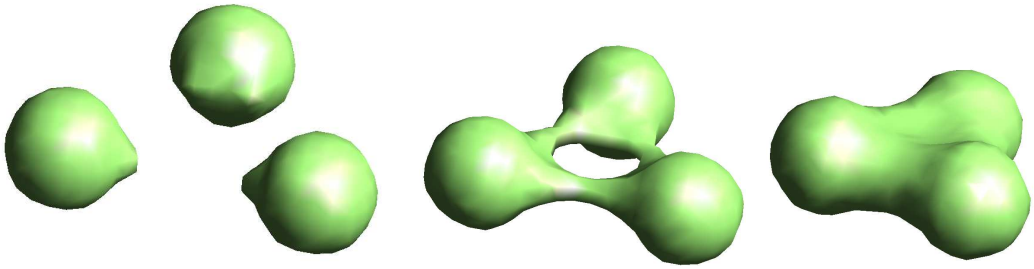
model as shown in figure 3.5. Again, we get similar results compared to [18]. This case occurs when there are three intersecting SAS atoms, and can be enumerated by our grid.



(a) The toroidal patch is disjoint and there is no self intersection.

(b) As the atoms come closer, a well defined toroidal patch is created.

Figure 3.4: The solvent excluded surfaces of two atoms which come closer.



(a) The 2 concave patches are disjoint and there is no wrong intersection.

(b) As the atoms come closer, a well defined patch, similar to the approximations in [18] is created.

(c) At mutually closer distances, the topology changes and the center hole disappears.

Figure 3.5: The solvent excluded surfaces of three atoms which come closer.

3.2.5 Operations Supported by the Adaptive Grid

Surface atoms detection Surface atoms are defined as those within a certain distance from the Molecular Surface. To obtain these atoms, we first compute the Molecular Surface. Next we search locally around each atom to

find the distance of the atom from the surface. This operation is linear in the number of atoms and cubic in the resolution of the grid.

Population of skin region We define the skin region of one molecule as the region belonging to the probe as it rolls on the surface, and defined as Solvent Accessible Surface 2 Volume (V_{SAS2}). We define the skin implicitly as a set of spheres packing the region. The packing density is itself chosen to approximately equal the packing of the atoms belonging to the molecular surface. The region is defined over a trilinear grid in which the molecule is embedded. The grid spacing h is chosen to preserve the features of the molecule. Assuming that the interatomic distance is $\sim 1\mathring{A}$, we can use $h = 0.5\mathring{A}$. By finding the boundary vertices of the SAS, we can obtain potential centers for the skin spheres. A packing algorithm decides, based on the packing density required, if a potential center should contain an atom or not.

Area volume computations We use primal contouring to define the surface and volumes. The area under the surface is approximated by piecewise linear elements of the isocontour. The volume is approximated by the volume enclosed by that piecewise linear approximation. This cost is linear in the size of the grid.

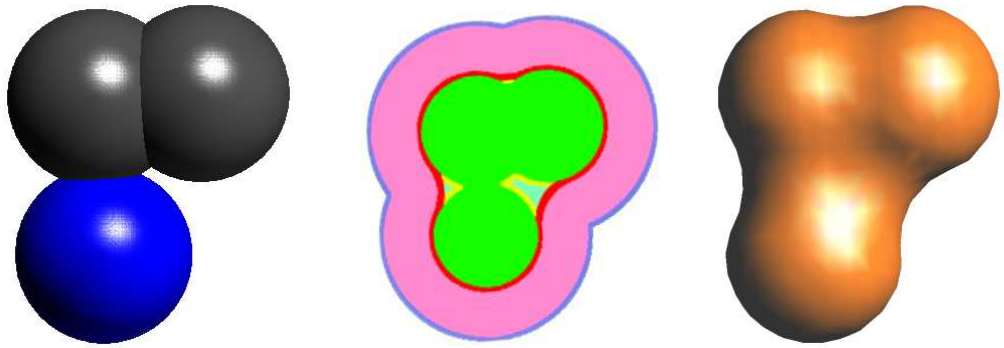
Curvature and normal computations These differential properties are computed using a two step process. Initially, when we propagate the distance

from the S_{SAS}, we also store whether the nearest patch is the intersection of one, two or more spheres. In each case, we can analytically provide the answer to the curvatures. For example, for a sphere with radius r, the Mean and Gaussian curvatures are $-1/r$ and $1/r^2$ respectively. In the second step, we compute the derivatives from the isocontour. At points where the two vary significantly, we choose to keep the value provided by the differencing scheme as the signed distance algorithm used is only an approximation.

3.2.6 Results

Region classification and construction of molecular surfaces Before we provide timing, geometric and functional properties and skin, surface regions, we present the results of our classification and signed distance function on a 3 atom model in figure 3.6(b). Using a relatively high resolution grid of 128^3, we classify grid points depending on the volume and surface it is part of, giving priorities of surface class over volumes and SES class over other surfaces. The figure is a 2D cross section of a volume rendering of the classified volume.

The Solvent Excluded Surface The solvent excluded surface is obtained as an isocontour with value equal to probe radius. In figure 3.7, we show colored visualizations of four different molecules.

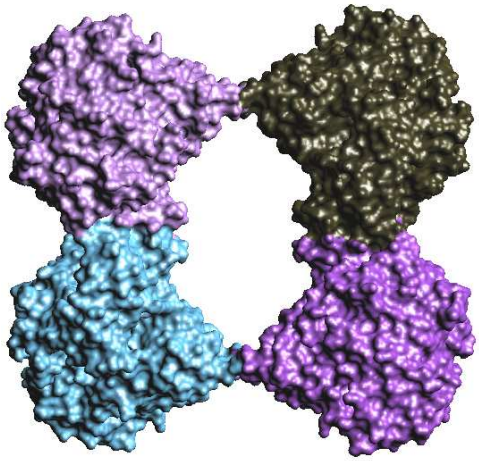


(a) Two carbons and a nitrogen atom are shown as spheres with radius given by their van der Waals radii.

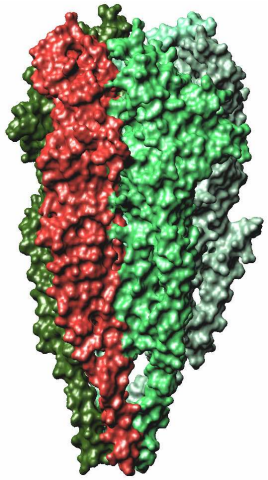
(b) A 2D cross section of the adaptive grid classification on a 3atom model.

(c) The isocontour of the function at 1.4Ågives the Richards surface.

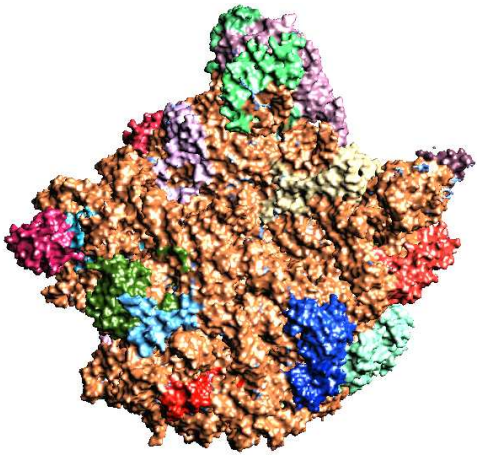
Figure 3.6: The cross section of the grid based SDF function for 3 atoms shows the following different surfaces and regions. S_{SAS}: dark blue, V_{SAS}: pink, S_{SES}: red, V_{SES}: light blue, S_{VDW}: yellow and V_{VDW}: green.



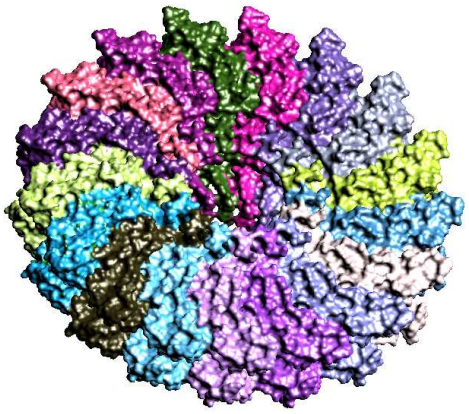
(a) An acetylcholine esterase (1C2B.pdb). It is shown in its tetramer form. Each unit, containing 4172 atoms each, is colored with a different color.



(b) The nicotinic acetylcholine receptor with over 14,000 atoms (2BG9.pdb). It has 5 chains, shown in different colors.



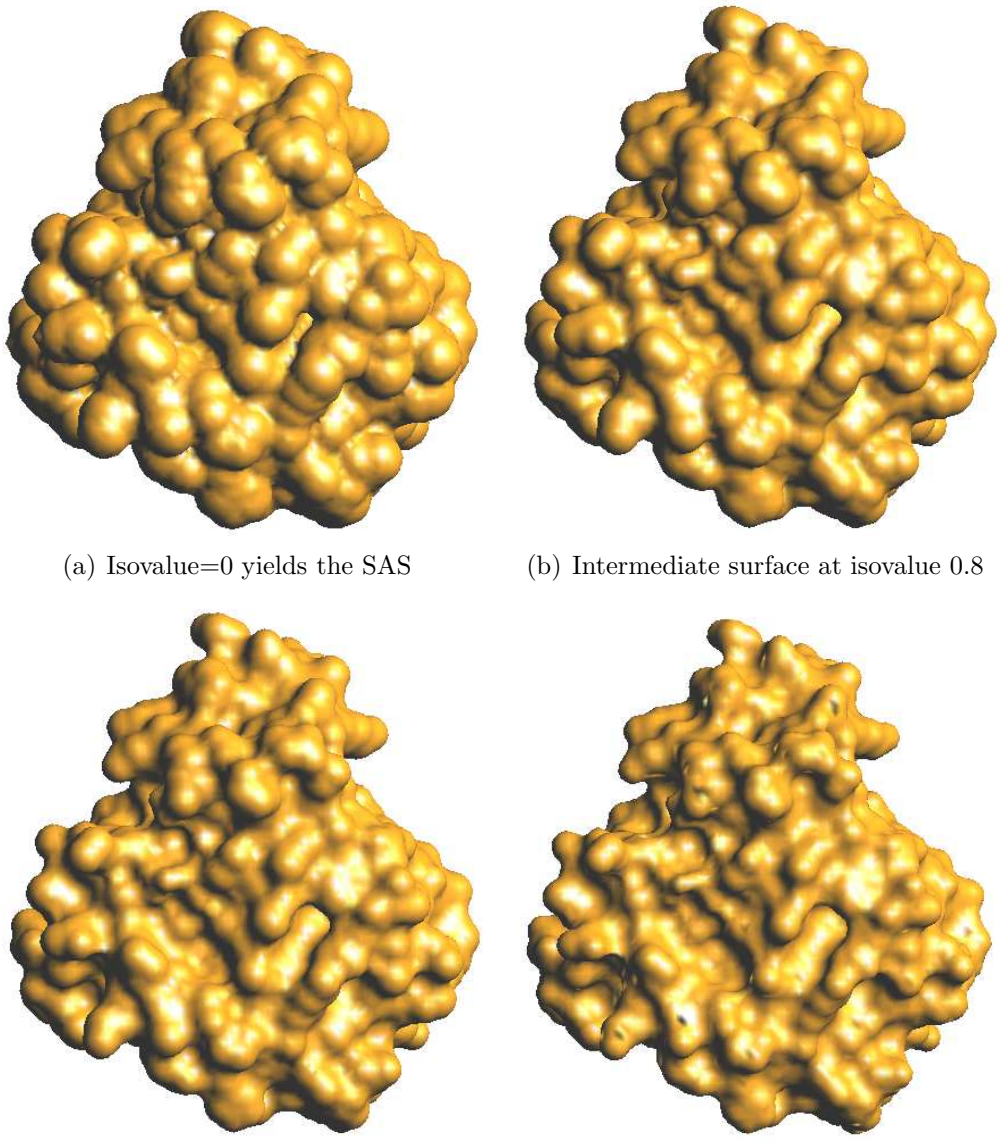
(c) The large ribosomal subunit (1JJ2.pdb) has almost 100,000 atoms. The RNA (in brown) and protein chains are shown.



(d) The tobacco mosaic virus, a helical virus (1EI7.pdb). The repeating subunits, each containing 2806 atoms, are shown for two rings.

Figure 3.7: The solvent excluded surfaces of four different molecules.

Family of surfaces In figure 3.8, we show four different surfaces computed from the adaptive grid, at four different isovalues. The myoglobin molecule, 101M.pdb, is used as a test case. At a distance of 0, we get the SAS surface, which is the union of spheres model, with each atom represented as a sphere with radius equal to the sum of its radius and a probe radius. In this example, we used a probe radius of 1.4Å. As we change the isocontour from 0 to 1.4Å, we get a smooth deformation of the SAS surface to the SES surface, as shown in the different figures. Since we are interested in the SES, we do not compute further in practise, but in theory, higher isovalues will take us closer to the van der Waals surface. This example shows the utility of our method as a volume exclusion function for computing electrostatics, which needs a smooth decay at the SES boundary.



(a) Isovalue=0 yields the SAS

(b) Intermediate surface at isovalue 0.8

(c) Intermediate surface at isovalue 1.1

(d) At isovalue 1.4 (probe radius), we obtain the SES.

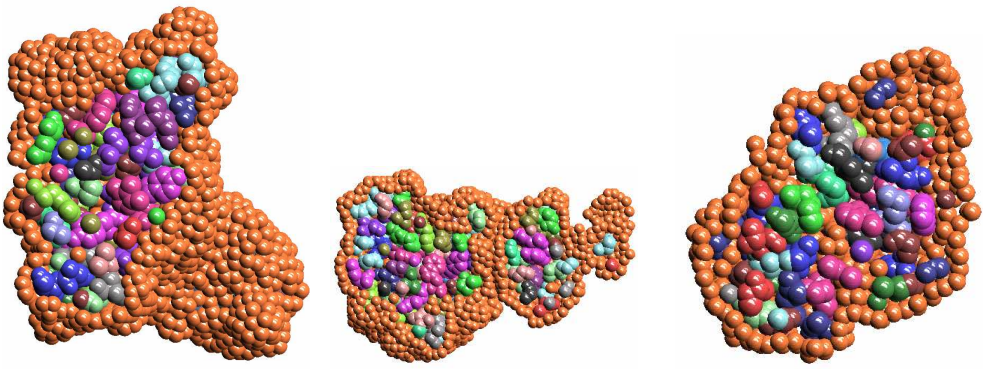
Figure 3.8: Our signed distance function based definition yields a family of surfaces which we can extract using a novel adaptive grid based algorithm.

PDB Id	Number of atoms	time (64^3)	time (128^3)
3sgb	1912	11	85
1brc	2197	6	58
2ptc	2243	6	53
2kai	2267	7	74
3tpi	2313	6	54
1tab	2387	9	72
1ppf	2520	7	63
4cpa	2739	10	85
1mkw	4844	8	60

Table 3.1: Times (in seconds) taken to compute the adaptive grid based surfaces and volume regions for different initial grids which are adaptively subdivided to a depth of 3.

Timing The cost of the algorithm depends on the depth of the adaptive grid, the resolution of the initial base grid and the size of the molecule. In table 3.1, we provide the time taken to compute the properties on the grid, including surfaces and demarking volumetric regions for different molecules and grid sizes. As the number of atoms increase, the time taken increases, but the fixed output grid size reduces the number of relevant search points within the SAS and VDW regions. Hence there is no direct correlation seen between the two. If the grid resolution can be chosen depending on the molecule size, then the time would increase monotonically with the number of atoms for molecules with similar distribution of atoms (say for a set of globular proteins).

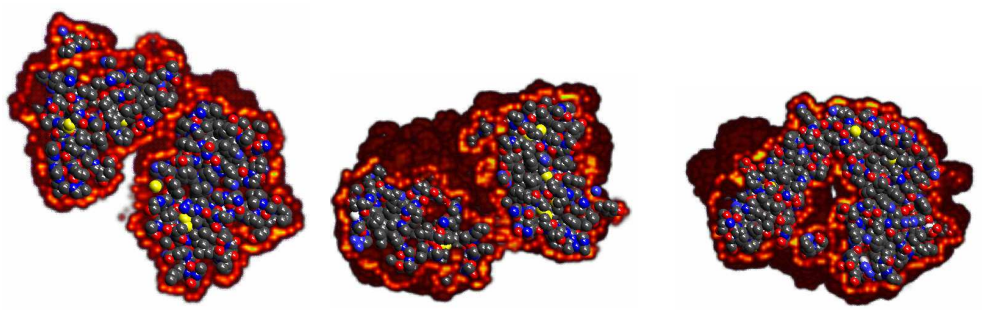
Surface atoms detection The surface atoms of three proteins from the complexes, anti-idiotypic fab (1IAI.pdb), hemagglutinin (2VIR.pdb) and bob-white quail lysosyme (1BQL.pdb) are visualized in figure 3.9. The interior atoms are colored by the residue they belong to while the outer surface atoms all have an orange color. We show a cutoff of the three molecules to reveal the surface and interior.



(a) Hemagglutinin (b) Anti-Idiotypic Fab (c) Bob-white quail lysosyme

Figure 3.9: Surface atoms of three proteins shown in orange over the interior atoms which are colored by their residue type.

SAS^2 skin region construction From the same above three complexes (1iai,2vir and 1bql), we extract the second protein and compute the skin regions (see figure 3.10) defined by the volume where the probe is present and touching the molecule. This region is used later in docking as it represents a volume where the interface atoms from the docking protein have a high probability of being present.



(a) Immunoglobulin from influenza virus

(b) Idiotypic fab of virus neutralizing antibody

(c) Hylel 5 Fab

Figure 3.10: Complementary skin region of three proteins shown in red over the atoms which are colored by their type.

Geometric and Functional properties We will present geometric and functional properties in section 3.4, where we compare this algorithm with others.

3.3 Fast Radial Basis Function based Molecular Surface Computation

The electron density and shape are used in a similar sense in the literature with respect to molecular surface modeling. The electron density of an atom at a point $\mathbf{x}$ is represented as a Gaussian function: $f(\mathbf{x}) = e^{\beta(\frac{|\mathbf{x}-\mathbf{c}|^2}{r^2}-1)}$ where $\mathbf{c}, r$ are the center and radius of the atom. If we consider the function value of 1, we see that it is satisfied at the surface of the sphere $(\mathbf{x} : |\mathbf{x}-\mathbf{c}| = r)$. Using this model, the electron density of a protein with M atoms at $\mathbf{x}$ is just a summation of Gaussians:

$$f(\mathbf{x}) = \sum_{i=0}^{M-1} e^{\beta(\frac{|\mathbf{x}-\mathbf{c}_i|^2}{r_i^2}-1)} \tag{3.1}$$

where β is a parameter used to control the rate of decay of the Gaussian and known as the *blobbiness* of the Gaussian. Isosurfaces of this function with isovalue 1 are extracted using traditional isosurfacing methods like primal/dual marching cubes. In [143] $\beta = -2.3, isovalue = 1$ is provided as a good approximation to the molecular surface. Through correspondence with Dr Wah Chiu's group, and from EMAN, we also have the following parameters for the Gaussians: The Gaussian is weighted by the number of electrons τ_i for the i^{th} atom. The resolution is taken as the distance in Åwhere the Gaussian function decays to either 0.5 or 1/e of the peak. $f(\mathbf{x}) = \sum_{i=0}^{M-1} \tau_i e^{-a|\mathbf{x}-\mathbf{c}_i|},\ a = \frac{\log 2}{res^2},\ or\ a = \frac{1}{res^2}$ Another method uses the Fourier domain: $\mathcal{F}(e^{-ax^2})(k) = e^{-\pi^2 k^2/a},\ a = \frac{\log 2\pi^2}{res^2},\ or\ a = \frac{\pi^2}{res^2}$

3.3.1 Related Work

Direct summation is often used, especially for truncated Gaussian kernels with (relatively) small width. If we have N output points and M Gaussians, then we can compute the sum at all the points in O(NM) time and O(M+N) memory. If the rate of decay of the Gaussian is high, then we can use a truncated Gaussian and update only around it. Consider a width of w for a truncated Gaussian. Using local summations, we get a computation cost of Mw^3. For grids (where N is large, typically 128^3 to 256^3), and slow decay Gaussians, this operation can be expensive. In the case of non decaying

kernels like thin plate splines and uniform output grids, this operation can be impractical. For the truncated summation, let the maximum extent of the kernel be l_{min}. If r_{max} is the maximum radius of the atoms and ϵ is the user allowed error, then, for M atoms: $l_{min} \geq \sqrt{r_{max}^2\left(\frac{\log(\epsilon/M)}{\beta}+1\right)}$ For non equispaced output points, we need to construct a space subdivision where we insert both the kernel centers and the output points. Using the grid, we can obtain all kernels influencing a certain point (or vice versa).

Since the sum of Gaussian can be expressed as a convolution, we can employ the Fast Fourier transform to obtain the summation: $f(\mathbf{x}) = G \otimes \sum_{i=0}^{M-1} \delta(\mathbf{x}-\mathbf{c}_i) \approx \mathcal{IFFT}(\mathcal{FFT}(G) \times \mathcal{FFT}(\sum_{i=0}^{M-1} \lfloor \mathbf{c}_i + [0.5, 0.5, 0.5] \rfloor))$. This is the traditional choice of algorithm used in practise due to the ease of implementation and speedup provided by the FFT. The transform for the atom centers set is performed over uniform grids, with the atom centers located to the closest grid points (reason for the approximation). On the other hand, the Gaussian functions Fourier transform can be done analytically. For a uniform grid of N^3 points, this summation takes $O(N^3 \log N + M)$ cost. If we consider a few number of output points, then this technique, which computes the output on a uniform grid, is inefficient.

In our technique, we concentrate on using Non equispaced Fast Fourier Transform algorithms to compute Fourier Coefficients and provide error bounds to the users. This method has numerous advantages, as shall be made clearer later.

3.3.2 Algorithm

Any periodic bounded function can be expanded as a Fourier series. For example, a periodic function in $[-1/2, 1/2]$ can be expressed as $f(x) = \sum_{j=-\infty}^{\infty} \omega_j e^{2\pi i j x}$ where the coefficients $\omega_j = \int_{-1/2}^{1/2} f(x) e^{-2\pi i j x} dx$ Let g be the truncated Gaussian function located at each atom. The truncation does not change the problem being solved as we are interested in a finite domain. $g(\mathbf{x} - \mathbf{x_c}) = e^{\beta(\frac{|\mathbf{x}-\mathbf{x}_c|^2}{r^2}-1)}$ is defined only in the domain of atom centers to make it a compactly supported function. The rate of decay $\beta < 0$, r is the radius. Let I_n denote a 3D grid of indices: $\{k : [-n/2..n/2)^3, k \in I\}$. Let us expand the Gaussian function in its fourier series form:

$$g(\mathbf{x} - \mathbf{x}_k) = \sum_{\boldsymbol{\omega} \in I_\infty} G_{\boldsymbol{\omega}} e^{2\pi i (\mathbf{x}-\mathbf{x}_k).\boldsymbol{\omega}}$$

The coefficients G_ω can be computed numerically. In [135], they obtain the coefficients by performing a $\mathcal{FFT}$ of a Gaussian over a grid of size n^3, which can be both inefficient and inaccurate. Substituting the expansion for the Gaussian , we get

$$f(\mathbf{x}) = \sum_{k=1}^{M} c^k \Big(\sum_{\boldsymbol{\omega} \in I_\infty} G_{\boldsymbol{\omega}} e^{2\pi i (\mathbf{x}-\mathbf{x}_k).\boldsymbol{\omega}} \Big) \approx \sum_{k=1}^{M} c^k \Big(\sum_{\boldsymbol{\omega} \in I_n} G_{\boldsymbol{\omega}} e^{2\pi i (\mathbf{x}-\mathbf{x}_k).\boldsymbol{\omega}} \Big)$$

$$f(\mathbf{x}) \approx \sum_{\boldsymbol{\omega} \in I_n} G_{\boldsymbol{\omega}} e^{2\pi i \mathbf{x}.\boldsymbol{\omega}} \sum_{k=1}^{M} c^k e^{-2\pi i \mathbf{x}_k.\boldsymbol{\omega}}$$

The second sum is the Discrete Fourier transform of the sum of atom centers. Let us denote these coefficients by $C_{\boldsymbol{\omega}}$.

$$f(\mathbf{x}) \approx \sum_{\boldsymbol{\omega} \in I_n} G_{\boldsymbol{\omega}} C_{\boldsymbol{\omega}} e^{2\pi i \mathbf{x}.\boldsymbol{\omega}} = \sum_{\boldsymbol{\omega} \in I_n} F_{\boldsymbol{\omega}} e^{2\pi i \mathbf{x}.\boldsymbol{\omega}} \tag{3.2}$$

In Appendix A, we provide error analysis and computational cost analysis for the algorithm. In particular, we obtain, for M atoms, N output points, n Fourier coefficients and a accuracy requirement ϵ is:

Lemma For tensor product kernels with Fourier coefficients K_ω, the number of coefficients n needed is at most:
$n = min(\hat{n}) : \sum_{\omega \in I_{\hat{n}}} (K_\omega)^2 \geq \frac{V}{2\pi} - \frac{M min_j(|c_j|^2)V}{(||c||_1)^2}(\frac{\epsilon}{3})^2$, where V is the integral of the kernel from $(-0.5..0.5]^3$.

Lemma For tensor product kernels whose Fourier coefficients decay at least inversely with frequency, the number of coefficients n needed is $O(M^{1/3}\epsilon^{3/2})$.

Lemma The fourier coefficients of a Gaussian function e^{-Bx^2} decay as the inverse of the frequency ω:
$G_\omega \leq max(\frac{1}{2\pi\sqrt{\pi}}, \frac{1}{2\pi} erf(\frac{\pi}{\sqrt{B}}), \frac{3\sqrt{B}}{e\pi^{3/2}}, 4\sqrt{\frac{2}{\pi e}}, \frac{Be^{-(1+\pi^2/B)}}{\pi^4})\frac{1}{\omega}, \quad (\omega \geq 2)$. The truncation of the Gaussian can be expressed as convolution with a sync function in Fourier space. Hence the Fourier series coefficients of the truncated Gaussian function can be now written as
$\int_{-\infty}^{\infty} \sqrt{\frac{\pi}{B}} e^{-\pi^2 t^2/B} sin(2\pi\omega)/(2\pi\omega - t) dt$. We then bound the sync function with a polynomial and integrate by parts to obtain the result.

Hence, we can summarize the algorithm as follows:

- n is computed from the error bounds and is $O(M^{3/2})$.
- $G_{\boldsymbol{\omega}}$ is computed numerically.
- $C_{\boldsymbol{\omega}}$ is computed approximately the using NFFT' algorithm.
- $f(\mathbf{x})$ is computed using an Inverse Fourier Transform for uniform grids and using the NFFT algorithm for non uniformly distributed output points.

The computational and memory cost analysis is given below.

- Estimation of the number of coefficients n required takes $O(M)$ time. $G_{\boldsymbol{\omega}}$ are computed numerically in 1D.
- Given NFFT' sampling parameters $\alpha \approx 2, m \approx 3$, the cost of computing $C_{\boldsymbol{\omega}}$ is $O(Mm^3 + n^3 \log n)$ and $O(M + (\alpha n)^3)$ space.
- The Inverse Fourier Transform takes $O(N^3 \log N)$ time and $O(N^3)$ memory. To expand p output points only, the computational cost is $O((\alpha n)^3 \log n) + mp$ and takes $O((\alpha n)^3 \log n + p)$ space.

3.3.3 Fast Update

Dynamic maintenance of molecular surfaces is an important requirement for surface construction algorithms due to the inherent flexibilities in

proteins. Eyal and Halperin in [54] provided algorithms to update parts of a moving molecule for the van der Waals surface. Dynamic maintenance and update with changes in probe radius for the explicitly computed NURBs patches of the SES is given in [16]. We provide methods to update the summation function and its derivatives due to the displacement of atom centers. This is made possible due to the linearity in equation (3.2) and the subsequent steps in the main algorithm. If we are given a representation of the object such that centers which move together are grouped, then we can take advantage of that distribution. This is especially true of proteins involved in protein-protein docking, which are large and exhibit domain motions and local side chain movements. Let us consider the grouping of M centers as follows

$$S = \{S_1, S_2, ..., S_{c(S)}\} \tag{3.3}$$

Since it is disjoint, we have that $M = c(S_1) + c(S_2) + .. + c(S_{c(S)})$. Let us introduce corresponding index sets J:

$$\begin{array}{ll} J^1_{m,n}(l) & \{j \in I_{S_1} : l - m \leq nw_j \leq l + m\} \\ J^2_{m,n}(l) & \{j \in I_{S_2} : l - m \leq nw_j \leq l + m\} \\ ... & ... \\ J^{c(S)}_{m,n}(l) & \{j \in I_{S_{c(S)}} : l - m \leq nw_j \leq l + m\} \end{array}$$

Using [135], equation (3.2) can now be further decomposed as a sum over a uniform grid. Hence, the sum from each domain is expressed as:

$$
\begin{aligned}
f_l &= \sum_{j \in J_{n,m}(l)} h_j \phi(w_j - \frac{l}{n}), l \in I_n \\
&= \sum_{j \in J^1_{n,m}(l)} h_j \phi(w_j - \frac{l}{n}) + ... + \sum_{j \in J^{c(S)}_{n,m}(l)} h_j \phi(w_j - \frac{l}{n}), l \in I_n \\
&= f^1_l + ... + f^{c(S)}_l
\end{aligned} \tag{3.4}
$$

The cost to perform this operation, if only one domain (say the d^{th}) moves is $\mathcal{O}((2m+1)^3 c(S_d))$. All the steps in the rest of the algorithm are also linear. Hence each stage can be broken up into $n(S)$ steps. Let the movement of a domain d affect N_d output points. Thus the total cost to update the function f when the domain d moves is $\mathcal{O}((2m+1)^3 c(S_d)) + \mathcal{O}(\alpha^3 M log(\alpha^3 M)) + \mathcal{O}((2m+1)^3 N_d)$ The cost is seen to be efficient compared to a brute force algorithm only if we are dealing with large kernels.

3.3.4 Results

We present the time taken to compute molecular surfaces based on the sum of Gaussians model using our fast summation method for two systems: The large ribosomal subunit (1JJ2.pdb), a large molecule containing 90403 atoms, and a relatively smaller myoglobin (101M.pdb), containing just over 1200 atoms. The molecules are blurred over uniform grids of size 128^3 and 512^3 respectively. For each molecule, we also compute the electron density with the same Gaussian parameters using a direct summation method and the FFT based technique to compare with. Two different resolutions are used: one at atomic and another at a 4Å. We also look at 1, 5 and 10 percent errors.

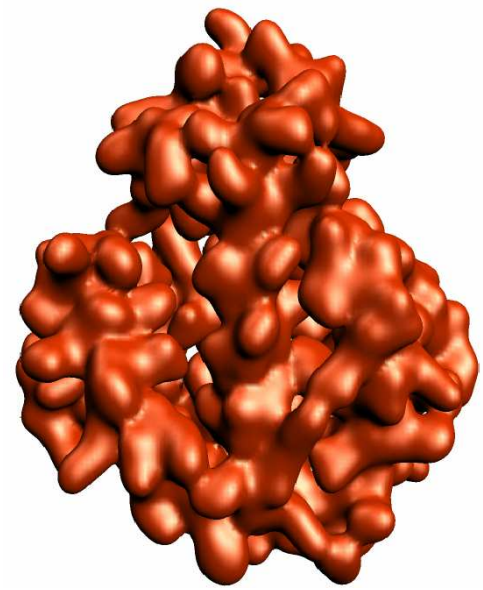
Timing Using a truncated Gaussian, (truncated where the value of the Gaussian is less than 10^{-3} of its peak), a direct sum of the kernels performs faster in the case of myoglobin than the ribosome, compared to the FFT technique. With an increase in the resolution required, the direct summation's time decreases as the cost is dependent on w^3, where w is the width of the truncated Gaussian. On the other hand, the FFT takes approximately the same time in both the forward and backward transforms, independent of the kernel parameters. Due to this reason, we only provide a single timing for the case of the FFT method. In our NFFT based algorithm, since for this case study, we are obtaining the result on a uniform grid, the back transform is independent of the size of the molecule and depends on the size of the output grid. Hence the time taken in our algorithm is at least more than half that of the time taken by the FFT method. Our algorithms time complexity is proportional to the number of atoms for the forward step. With a decrease in the resolution required, the time decreases, as shown by the complexity analysis. In table 3.2 and 3.3, the timings for the two models are given. In column (a), the forward time is presented, and, as expected, decreases as lesser accuracy is desired, or when we go to lower resolutions. It is always seen to be less than either the FFT or direct sum techniques. In column (b), the full time taken by our algorithm is shown. This value is lesser than the direct summation as we go to lower resolution, and may not be faster at high resolutions. The next column presents the actual L_2 percent error in the volumes and the last column the number of frequencies used. If we are interested in only a few output

points, or in non uniformly distributed output points, then in table 3.4, we present the times taken to compute the values. As expected, computing using a FFT to a uniform grid is faster than the NFFT method.

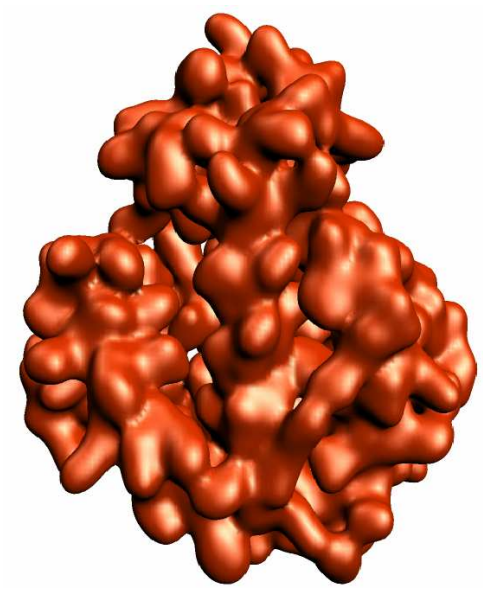
Error The error in the FFT method is due to the movement of atoms to grid locations and not recorded. The error in the direct summation is due to the truncation of the Gaussian kernels and again not recorded. We compare the volumes generated by our algorithm to the direct summation, assuming the direct summation has high accuracy. In particular, we compute the number of Fourier coefficients required to obtain a certain error. Tables 3.2 and 3.3 show the relation between error and the number of coefficients required.

Visual comparison In figure 3.11, myoglobin and the large ribosomal subunits are computed using the direct sum and our algorithm, allowing for 10% error. At the isovalues we are interested in, the ringing artifacts due to truncation of Fourier coefficients are not seen and high visual accuracy is preserved.

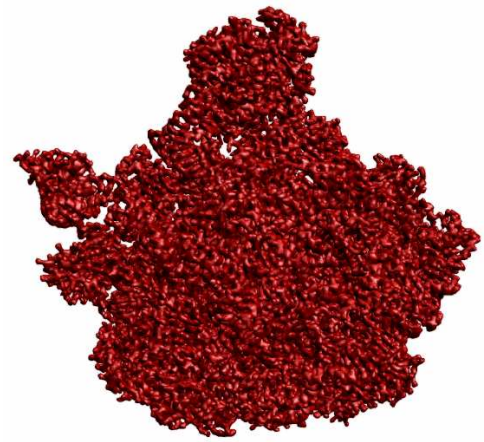
Results from the surface construction algorithms are compared using geometric properties. In each case, we compare the values to either exact known analytical solutions, or output from standard software. Four different surfaces are constructed:



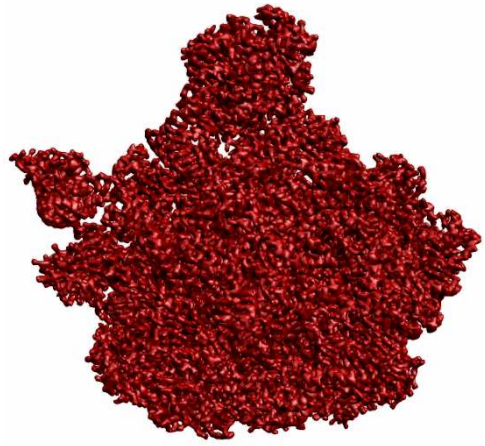
(a) Electron density of myoglobin molecule.



(b) Approximate within 10% error.



(c) Electron density of large ribosomal subunit.



(d) Approximate within 10% error.

Figure 3.11: Comparison of the fast blurring algorithm with the exact volumes. We present isosurface and volume renderings of two molecules: myoglobin (101M.pdb, 1221 atoms, 128^3 grid, at $1,4\mathring{A}$ resolution) and the large ribosomal subunit (1JJ2.pdb, 90403 atoms, 512^3 grid, at $1,6\mathring{A}$ resolution). Even at a 10% error, very high visual accuracy is present. Refer to tables 3.2 and 3.3 for timing results and other parameters.

Myoglobin (101M.pdb), 1221 atoms, $48\mathring{A} \times 40\mathring{A} \times 40\mathring{A}$, opt. grid size $=128^3$
FFT method time: 16.509123

	Res=1				Res=4			
Direct	6.730830				14.733616			
	a	b	c	d	a	b	c	d
$F.B., \epsilon \approx 1\%$	4.08	12.26	1.08	96	1.33	8.76	1.13	44
$F.B., \epsilon \approx 5\%$	3.01	11.49	4.74	76	1.17	8.54	5.08	32
$F.B., \epsilon \approx 10\%$	1.69	9.87	9.30	64	1.13	8.50	10.37	24

Table 3.2: Timing (in seconds) and errors (L_2 percent) for fast summation of 1221 gaussian kernels to a $128 \times 128 \times 128$ uniform grid. The entries contain the timing in seconds and the actual error for different resolutions. $m = 3$ was used for the interpolating functions in each case. The notations are: a: Forward time in seconds, b: Full time in seconds, c: Actual error, d: αM. Since we perform the blurring in frequency space, our method would be much faster than the others at such low frequencies. The quadratic algorithms gaussian was clamped when it reduced to 10^{-3} of its peak.

Large ribosomal subunit (1JJ2.pdb), 90403 atoms, $221\mathring{A} \times 221\mathring{A} \times 175\mathring{A}$, opt. grid size $=512^3$
FFT method time: 1425.239929

	Res=1				Res=6			
Direct	128.577673				5844.995279			
	a	b	c	d	a	b	c	d
$F.B., \epsilon \approx 1\%$	245.53	983.21	0.98	200	7.849469	760.38	0.94	60
$F.B., \epsilon \approx 5\%$	98.22	803.31	4.84	150	4.05421	743.42	4.78	40
$F.B., \epsilon \approx 10\%$	71.79	790.50	10.30	130	3.211702	730.82	10.22	32

Table 3.3: Timing (in seconds) and errors (L_2 percent) for fast summation of 90403 gaussian kernels to a $512 \times 512 \times 512$ uniform grid. The entries contain the timing in seconds and the actual error for different resolutions. $m = 3$ was used for the interpolating functions in each case. The notations are: a: Forward time in seconds, b: Full time in seconds, c: Actual error, d: αM. Since we perform the blurring in frequency space, our method would be much faster than the others at such low frequencies. The quadratic algorithms gaussian was clamped when it reduced to 10^{-3} of its peak. We used a larger resolution range as this is a relatively large molecule.

Number of output centers	time in seconds
1000	0.078514
10000	0.787210
100000	7.867189
1000000	77.809066

Table 3.4: Timing and errors for fast summation of gaussian kernels at different number of non-uniformly spaced output points, using $m = 3, \alpha = 2$ points. The time reported is the time it takes to compute the function given the sum of b-splines grid representation. We used the large ribosomal subunit electron density blurring at $1\mathring{A}$ resolution and 10% error. Please refer to table 3.3 for the time taken to precompute the grid at different errors and resolution.

3.4 Comparison of Molecular Surfaces

Results from the surface construction algorithms are compared using geometric properties. In each case, we compare the values to either exact known analytical solutions, or output from standard software. Four different surfaces are constructed:

1. Solvent excluded surface, constructed as an isocontour with isovalue 1.4 from our adaptive grid.

2. A sum of Gaussians, computed using a discrete set of radii, with rate of decays (B) of:

   (a) $\mathbf{B = -2.3}$: This value is used in the literature as one which gives a good approximation of the volume enclosed [143].

   (b) $\mathbf{B = -1.0}$: By reducing the value of B, the Gaussian is made smoother, leading to a lower resolution map.

   (c) $\mathbf{B = -3.0}$: We look at a sharper Gaussian to study the trend of properties as we move about the standard value of -2.3.

Datasets: We use a set of 71 complexes. Properties are computed and compared of both the complex and its two individual proteins. Hence, overall, we use 213 proteins in our dataset. The complex's Id in the figures is appended with a 'C', while the two proteins in it are appended with '1' and '2'.

3.4.1 Geometric Comparison

The geometry of the surface plays an important role in its structural and functional properties. Two of the geometric properties we consider are areas and volumes:

- **Area of the surface:** The area of the surface is computed using a simple approximation by summing the area of triangles contained in the isocontours. MSMS [147] computes an analytical solution to the solvent excluded surface area. We compare our results with theirs.
- **Volume enclosed by the surface:** The volume enclosed is computed by summing partial or full contributions of voxels in the grid. If $n, n \leq 8$ grid corners are included in the volume, then $n/8$ of the voxels volume is counted.

In figures 3.12 and 3.13, we plot the areas (in $\mathring{A}^2$) of five different surfaces (our four and the numerical value computed by running MSMS) with the analytical solvent excluded surface area computed by running MSMS. While using MSMS, we set the probe radius to 1.4 and do not change the default values of grid spacing etc. From the figures, we see that the adaptive grid and the MSMS numerical values are just slightly lower than the analytical value. Among the sum of Gaussian surfaces, a rate of decay of 1.0, which provides a smooth surface was seen to be a good approximation, while sharper Gaussians tends to increase the surface area due to the increase in 'bumps' and 'valleys'.

Volumes (in Å^3) enclosed in the solvent excluded surface are compared in figures 3.14 and 3.15. The adaptive grid algorithm and sums of sharper Gaussians, including rates of decay of $-2.3, -3$, are seen to be close to the numerical value computed by MSMS. Unfortunately, the volume under the smoother surface defined by a Gaussian sum with decay rate of -1.0 is seen to overestimate the volume, as expected. Proteins in all the above results are ordered by their analytical solvent excluded area, as computed by MSMS.

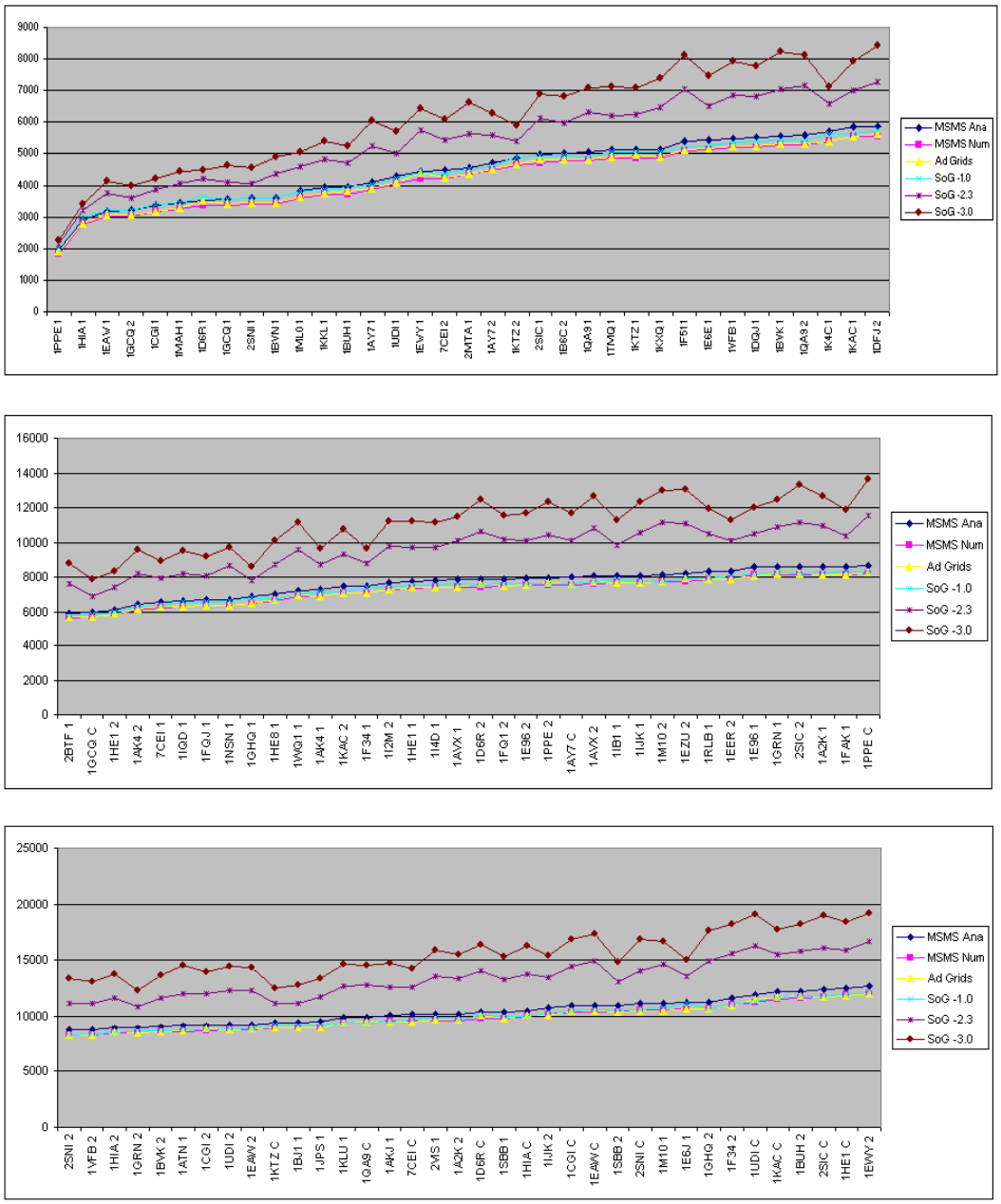


Figure 3.12: Area comparisons 1,2,3: We compare the areas (in Å^2) of surfaces of molecules computed using our adaptive grid algorithm (yellow), the analytical surface area by MSMS (dark blue), MSMS numerical surface area (pink) and the sum of Gaussians method with three different rates of decay (light blue, purple and brown).

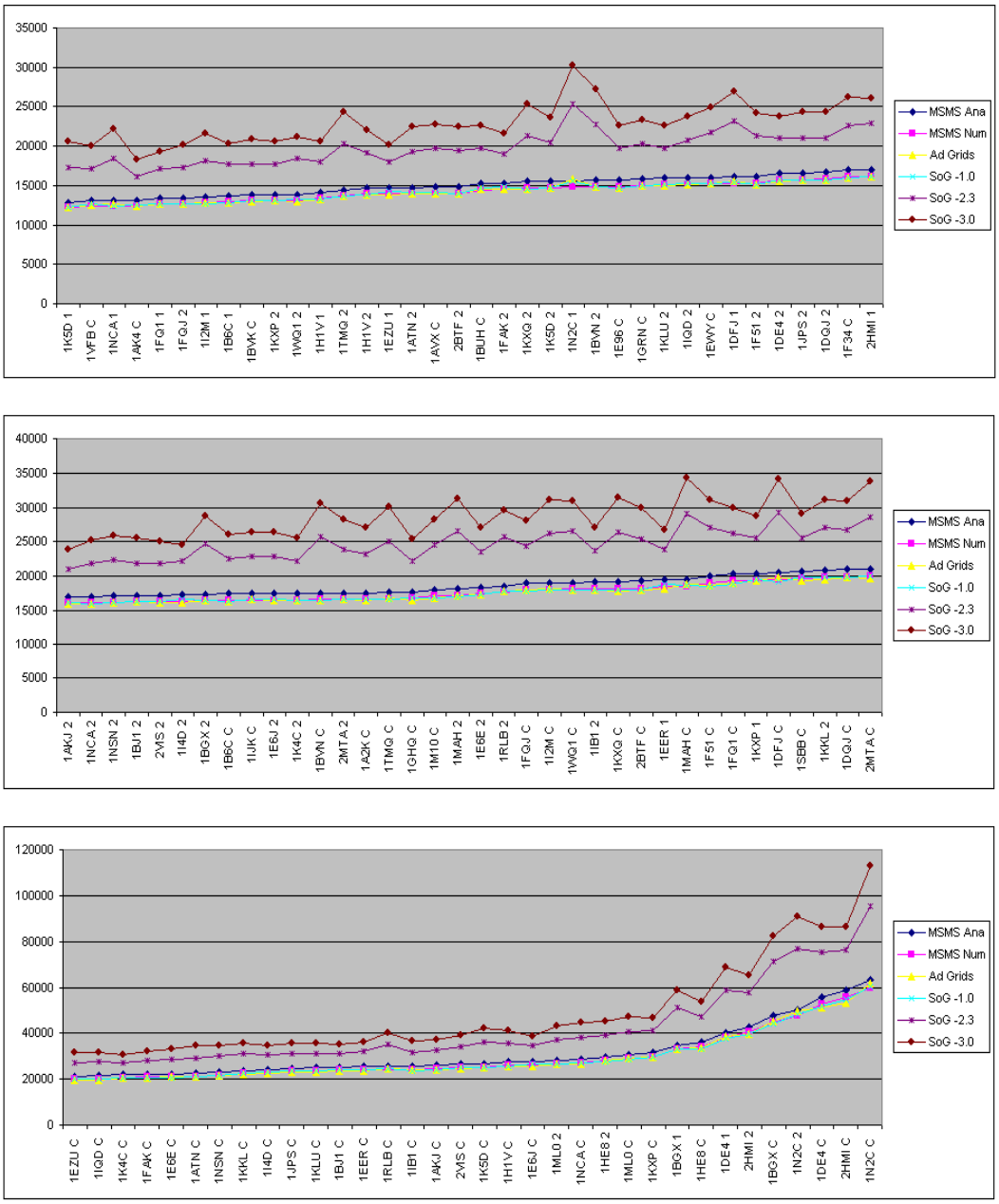


Figure 3.13: Area comparisons 4,5,6: We compare the areas (in Å^2) of surfaces of molecules computed using our adaptive grid algorithm (yellow), the analytical surface area by MSMS (dark blue), MSMS numerical surface area (pink) and the sum of Gaussians method with three different rates of decay (light blue, purple and brown).

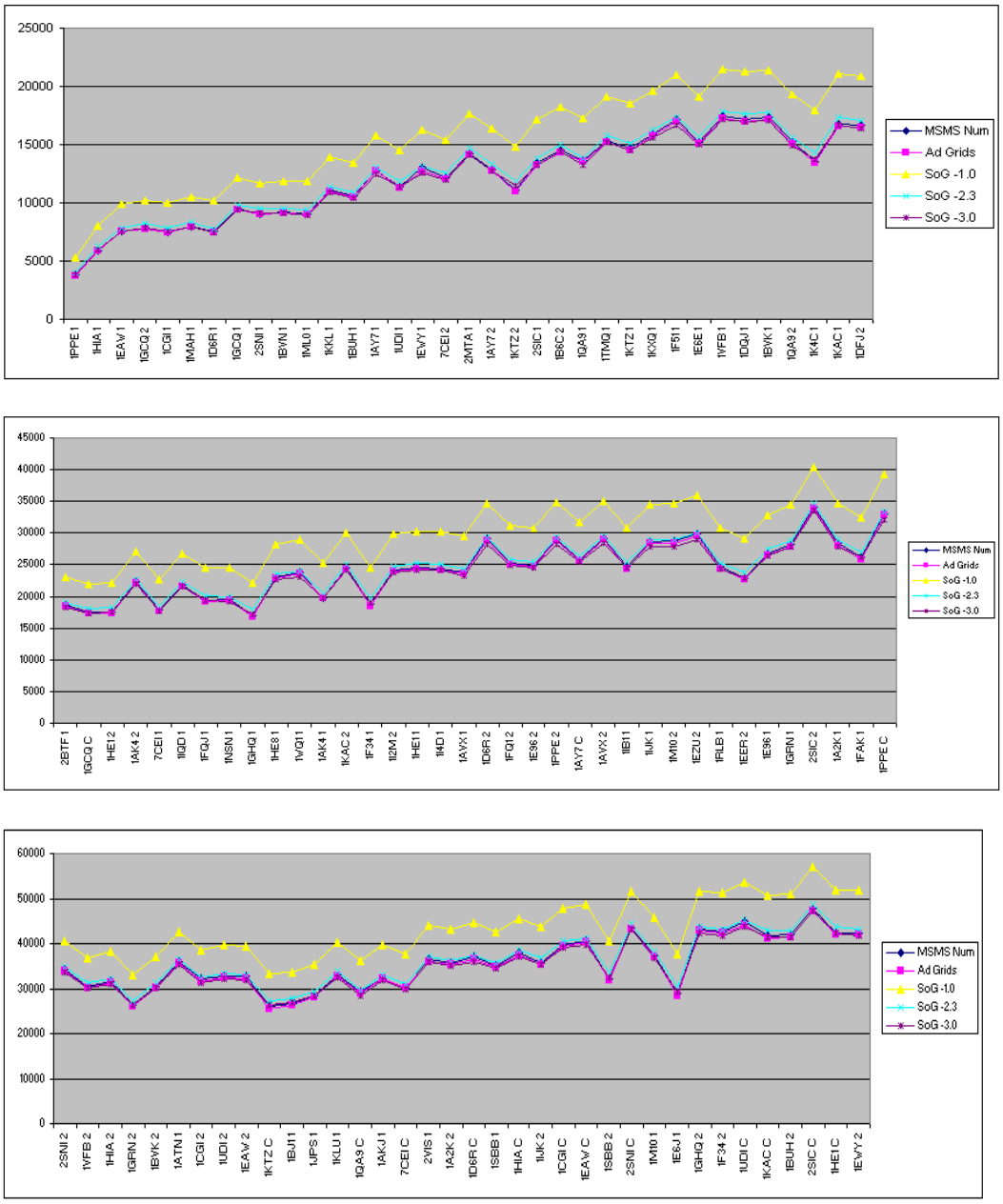


Figure 3.14: Volume comparisons 1,2,3: We compare the volumes (in $\mathring{A}^3$) enclosed by surfaces of molecules computed using our adaptive grid algorithm (pink), MSMS numerical volume (dark blue) and the sum of Gaussians method with three different rates of decay (yellow, light blue and brown).

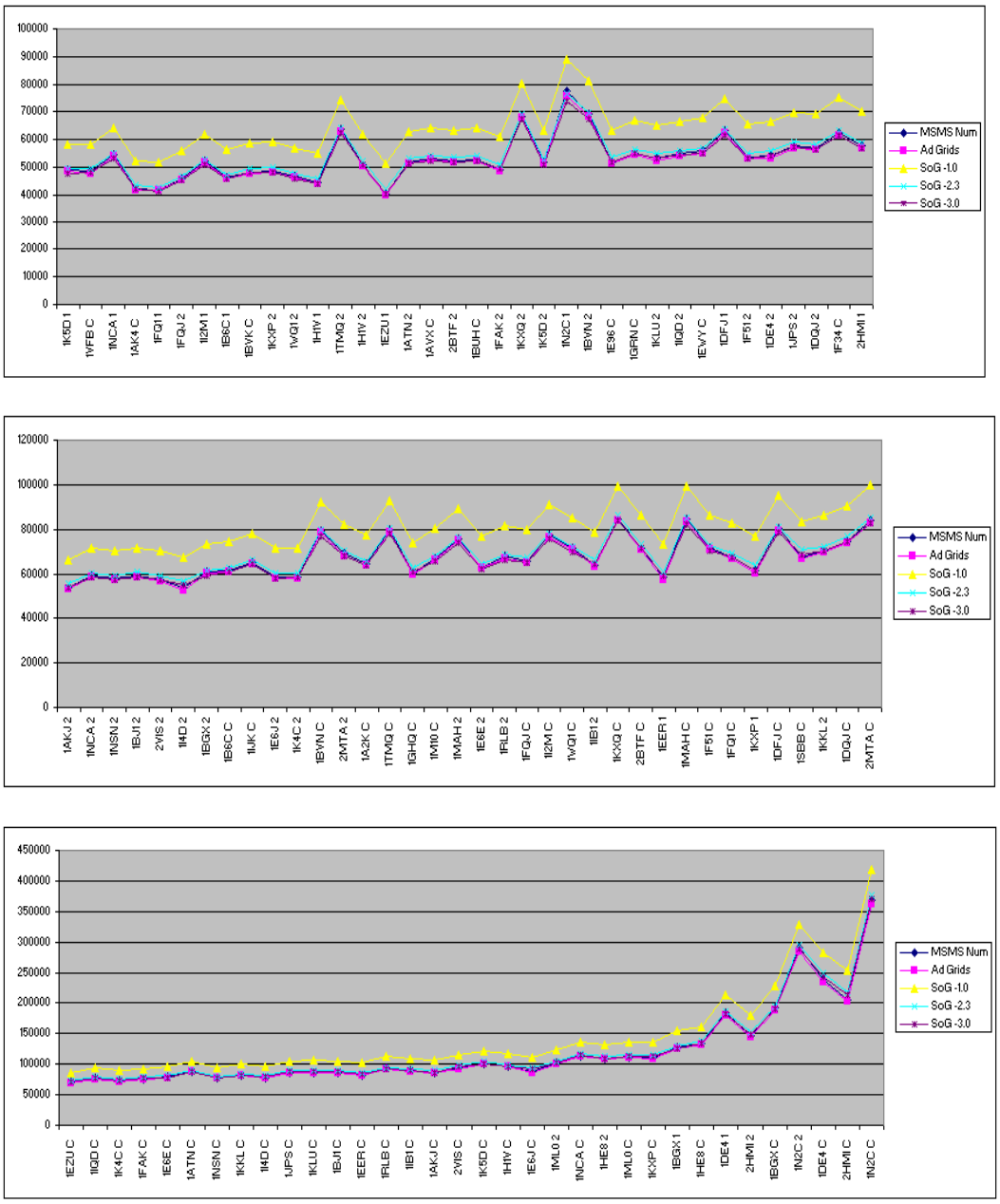


Figure 3.15: Volume comparisons 4,5,6: We compare the volumes (in $\mathring{A}^3$) enclosed by surfaces of molecules computed using our adaptive grid algorithm (pink), MSMS numerical volume (dark blue) and the sum of Gaussians method with three different rates of decay (yellow, light blue and brown).

3.5 Summary

The Richards model for molecular surfaces is computed using a signed distance function on an adaptive grid, which also yields additional surfaces, regions and properties of the molecule. An implicit Sum of Gaussians function is computed using a fast summation algorithm based on non-equispaced fast Fourier transform algorithm. The implicit surface supports a multiresolution representation through its rate of decay parameter. We compare geometric properties of the two surfaces and compare with MSMS, a well known software. This radial basis function representation of the molecular electron density and adaptive grid based definitions of surfaces and regions is used in our fast docking algorithm.

Chapter 4

Protein-protein Docking

We introduce a novel grid-free protein-protein docking algorithm using non equispaced Fast Fourier techniques.

4.1 Docking Model Specification

Consider two proteins with M_1, M_2 atoms respectively. Different structural and functional properties of the two proteins interact in determining its docking site and score. Let these functions, formally known as affinity functions be $A_{1,i}, A_{2,i}, i \in 1..N$. To compute the interaction between two affinity functions of the proteins, a scoring function is introduced. Let us denote by $f_{1,i}, f_{2,i}$, the pair of scoring functions for the i^{th} affinity function. Let T, Δ be traditional 3D translational and rotational operators. If the user considers a potential docking site as one where the overlap potential is over a threshold τ, then the protein-protein docking solution is expressed as the set of triplets:

$$\{(\mathbf{t}, \mathbf{r}, s) : (s = Re(\sum_{k=1}^{N}(\int_{\mathbf{x}} f_{1,k}(\mathbf{x}) T_{\mathbf{t}}(\Delta_{\mathbf{r}}(f_{2,k}(\mathbf{x})))d\mathbf{x}))) \geq \tau \forall (\mathbf{t}, \mathbf{r})\} \qquad (4.1)$$

4.2 Affinity Functions

In the first stage of protein-protein docking, shape complementarity, electrostatics, desolvation energy, and hydrogen bonding are commonly used affinity functions. The interactions of these affinity functions are captured in the scoring function to rank complexes. The scoring is made more accurate and thus becomes more computationally expensive in the latter stages (where perhaps dynamics is used for search). The binding coefficient can be derived knowing the change in free energy in solution, $\Delta G_{binding,solution}$ as $\Delta G_{binding,vacuo} + \Delta G_{solvation(EI)} - \Delta G_{solvation(E+I)}$. (see AutoDock [127].) The Gibbs free energy change, written as the change in enthalpy and the product of absolute temperature times the entropy is used to rank docked solutions. The change in enthalpy is given by the change in the terms: the internal energy changes in the individual proteins and complex, and intermolecular enthalpy. The internal energy terms are given as: $E_b + E_\theta + E_\phi + E_{vdW} + E_{elec}$ while the solvation term is often expressed as a sum of solvent-solvent cavity term, a solute-solvent van der Waals term and a solute-solvent polarization term: $G_{cav} + G_{vdW} + G_{pol}$ [159]. The potential energy due to the van der Waals force of attraction between atoms and the strong repulsion of electron clouds as they come close is given by the well known 12-6 Lennard Jones approximation: $4\epsilon((\frac{\sigma}{r})^{12} - (\frac{\sigma}{r})^6)$, where the constants are chosen statistically. A similar 12-10 form is used to model hydrogen bonds energy. Due to the large contrast in dielectric between solvent and solute for the case of proteins in water and the large number of solvent molecules, electrostatics is computed

with a distance dependent dielectric and a distance cutoff. Energy in bonds, bond angle changes and torsional angles are computed as $\sum_{bonds} K_b^i(b_i - b_0^i)^2$, $\sum_{bondsangles} K_\theta^i(\theta_i - \theta_0^i)^2$ and $\sum_{torsionangles} K_\phi^i\{1 - cos[n^i(\phi_i - \phi_0^i)]\}$. See [110] for values for these constants.

4.2.1 Shape Complementarity

Shape complementarity is the most important affinity function used to score in protein-protein docking [52, 153]. The *double skin layer* approach is used to maximize the overlap of the surface of protein B with the complementary space of A. It was introduced in [168] for 2D, [84] for 3D, sped up using Fast Fourier Transforms in [91], and extended to complex space in [32]. Two *skin regions* are defined: 1). The *surface skin* of B, which is the density function of the set of surface atoms of B, and 2). The complementary region of A, defined by a *grown skin region*, by introducing a 1-layer potential on the surface of A. The atoms of A and the inner atoms of B form *core regions*. To maximize skin overlaps and to minimize overlaps of the cores, we assign positive imaginary weights to the core region and positive real weights to the skins we wish to maximally overlap. An integral of the superposition of the molecules has two real contributions: the core overlaps contribute negatively and the skin overlaps contribute positively. See [33] for an extension to shape complementarity to *pairwise shape complementarity*.

4.2.2 Electrostatics Interactions

Electrostatics plays an important role in protein-protein docking. It is used to improve the score obtained by shape complementarity. Due to partial charges on the proteins and the solvent, electrostatic forces are significant and influence both folding and docking. Formation of weak hydrogen bonds in interfaces are all due to these partial charges. The traditional way of computing this score is to evaluate the overlap of the electrostatic potential of one protein with the charges on another. There are various approximation involved to make this feasible in practise. Electrostatic potential is often computed using just a implicit model of solution, and in particular, the weaker linearized Poisson-Boltzmann equation. Partial charges are assigned using programs like CHARRM or AMBER to the atoms in protein. Like charges repel and opposite charges attract, and hence the scoring function is just the negative of the dot product of the two functions. The dot product at all overlap positions is just the convolution. Thus, if f_1 is the potential ϕ due to charges on the first protein and f_2 is a set of partial charges q_k of proteins on the second protein at positions $x_{2,k}$, the scoring function is simply $-f_1 \otimes f_2 = -\phi \otimes \sum_{k=1}^{M_1} q_k \delta(. - x_{2,k})$.

4.2.3 Scoring Using RBF Representations of Affinity Functions

The affinity functions are modeled as Radial Basis Functions (RBFs) to facilitate using Fourier transforms to efficiently solve the docking problem.

Molecule representation We use the sum of Gaussian's representation to model our proteins. An atom centered at $\mathbf{x}_c$, with a van der Waal's radius of r, is modeled as an isotropic Gaussian kernel: $g(\mathbf{x}) = e^{-\beta(\frac{(\mathbf{x}-\mathbf{x}_c)^2}{r^2}-1)}$. The decay rate of the kernel is controlled by β. A value of 2.3 is used in the literature [61] to approximate the solvent excluded surface at an isovalue of 1. By lowering this parameter, we can model molecules at lower resolutions [17].

Shape complementarity Since the symmetry is broken in the imaginary part of the integral (one contribution is due to atom - atom overlap and another from pseudo-atom - atom overlap), we currently do not use this value, although others in the literature assign this a 'smaller' negative potential. The weighted sum of Gaussians function definition of a molecule (of M atoms), with its associated *skin region* can be expressed as a sum of two functions: $f_{SC}(\mathbf{x}) = f^{Re}(\mathbf{x}) + f^{Im}(\mathbf{x}) = \sum_{k=1}^{M_{Re}} c^{Re} g(\mathbf{x}-\mathbf{x}_{Re,k}) + \sum_{k=1}^{M_{Im}} c^{Im} g(\mathbf{x}-\mathbf{x}_{Im,k}) = \sum_{k=1}^{M} c_k g(\mathbf{x}-\mathbf{x}_k)$. Here, g is the Gaussian function located at each atom (or pseudo atom) and (SC) stands for shape complementarity. The weights $\{c_k \in \{c^{Im}, c^{Re}\}, k = 1..M\}$ are either positive imaginary or positive real. We used our adaptive grid based algorithm §3.2 to construct the regions shown in figure 4.1.

Electrostatics scoring Similar to the procedure used for shape complementarity, Gabb et. al. [60] have shown how to introduce the electrostatics term. The first protein's electric potential is computed and matched against the charges in the other. A new affinity function f_E^1 is defined as $\sum_k q_k \frac{1}{E(\mathbf{x}-\mathbf{x}_k)(\mathbf{x}-\mathbf{x}_k)}$,

where $E(\mathbf{x})$ is the distance dependent dielectric constant [60]. The corresponding function for the second protein is $f_E^2 = \sum_{k=1}^{M_2} q_k \delta(\mathbf{x} - \mathbf{x}_k)$. In [32], they use these functions multiplied with a imaginary and a negative imaginary weight respectively. This is seen to have our desired radial basis format.

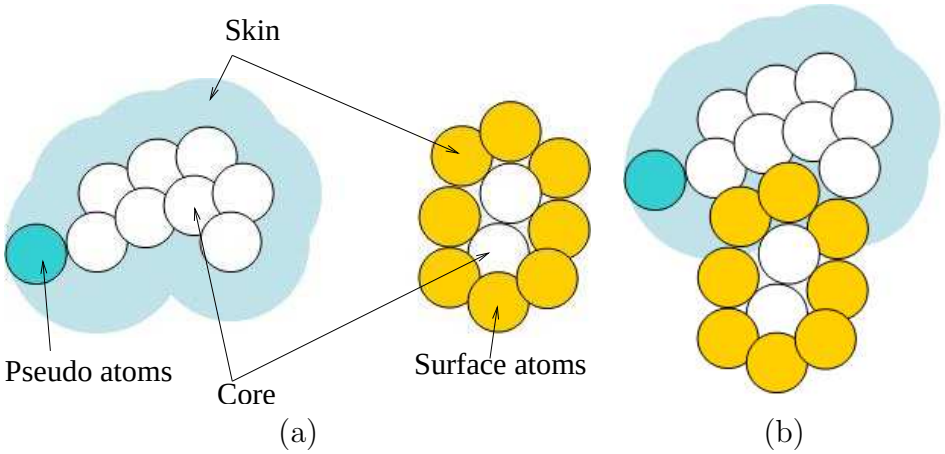


Figure 4.1: (a) Skin and Core regions for complementary space docking. Atoms are drawn as solid circles. The skins regions are colored while the core regions are white. (b) A possible docking of the molecules show a large overlap between the grown layer of the first and the surface atoms of the second.

4.3 Search Algorithm

Given affinity functions are Radial Basis Functions allows us model the docking search as convolutions, which can be sped up using a variety of approximation algorithms below. Traditional approaches involve using large grids and discrete analysis which loses accuracy and involves high computational and memory costs. The docking search is now expressed more succinctly as a set of three convolutions.

4.3.1 Translational Search

Given the scoring functions, we first express them in a form suitable for performing convolutions. First, we need to scale them to the region $[-0.5..0.5)^3$, and ensure that they are zero padded to allow for periodic overlaps. Given the centers $\mathbf{x}_i$, for all centers, and a width of the compactly supported kernel, the *diameter* w of the largest molecule is determined and the centers are scaled down to fit $[-0.25..0.25)^3$. Similarly, the kernel is also scaled in a similar fashion.

Given the new scaled and translated scoring functions f_1, f_2 and a rotation R, the docking score is given by the convolution integral:

$$\int_{\mathbf{y}\in\pi^3} f_1(\mathbf{y})(\Delta_R(f_2))(\mathbf{x}-\mathbf{y})d\mathbf{y},\ \forall\ \mathbf{x} \tag{4.2}$$

We solve equation (4.2) using Fourier series expansions. First, we express the integral as a uniform sum of compactly supported functions and provide an adaptive algorithm to search for regions where the scoring function exceeds the threshold provided by the user.

4.3.1.1 Fourier Series Expansions

Using the expansion in the fast summation algorithm in section 3.3, we express the convolution integral with approximate Fourier series. Our scoring function is seen to a periodic bounded function in $[-1/2, 1/2]$ and can hence can be expressed as: $f(x) = \sum_{j=-\infty}^{\infty} \omega_j e^{2\pi i j x}$, where the coefficients

$\omega_j = \int_{-1/2}^{1/2} f(x)e^{-2\pi ijx}dx$. The affinity function can be expressed as before:

$$f(\mathbf{x}) = \sum_{k=1}^{M} c_k g(\mathbf{x} - \mathbf{x}_k) = \sum_{\boldsymbol{\omega} \in I_\infty} G_{\boldsymbol{\omega}} C_{\boldsymbol{\omega}} e^{2\pi i \mathbf{x}.\boldsymbol{\omega}}$$

Similarly:

$$f(\mathbf{x} - \mathbf{y}) = \sum_{\boldsymbol{\omega} \in \mathbf{I}_\infty} \mathbf{G}_{\boldsymbol{\omega}} \mathbf{C}_{\boldsymbol{\omega}} \mathbf{e}^{\mathbf{2\pi i(x-y).\boldsymbol{\omega}}}$$

Expanding f_1, f_2 using the above series, the scoring integral in equation reduces to

$$\int_{\mathbf{y} \in \pi^3} f_1(\mathbf{y})(\Delta_R(f_2))(\mathbf{x} - \mathbf{y})d\mathbf{y},\ \forall\, \mathbf{x} =$$

$$\int_{\mathbf{y} \in \pi^3} \sum_{\boldsymbol{\omega}_1 \in I_\infty} G_{\boldsymbol{\omega}_1} C_{\boldsymbol{\omega}_1} e^{2\pi i \boldsymbol{y}.\boldsymbol{\omega}_1} \sum_{\boldsymbol{\omega}_2 \in I_\infty} G_{\boldsymbol{\omega}_2} C'_{\boldsymbol{\omega}_2} e^{2\pi i (\mathbf{x}-\mathbf{y}).\boldsymbol{\omega}_2} d\mathbf{y}$$

Since $\int_{-1/2}^{1/2} e^{2\pi iy(a-b)} = 1$ if $a = b$ and 0 otherwise, the integral is given as $\sum_{\boldsymbol{\omega} \in I_\infty} G^2_{\boldsymbol{\omega}} C_{\boldsymbol{\omega}} C'_{\boldsymbol{\omega}} e^{2\pi i \mathbf{x}.\boldsymbol{\omega}}$. The above Fourier series, for smooth, bounded inputs, is equal in the $||L||_2$ norm, and everywhere point-wise equal, except for a finite number of discontinuities.

4.3.1.2 Approximations

We make three approximations in computing the above coefficients. Since the truncated Gaussian is a decaying kernel, we choose to compute only

the first $(-n/2..n/2]^3$ Fourier coefficients. The parameter n is chosen to satisfy a user required accuracy in the docking profile. If we include electrostatics, the decay should be even slower, and hence, the same bounds derived for shape complementarity should be sufficient. The current analysis, though, is based on shape complementarity. The Fourier coefficients of the atoms centers, $C_{\boldsymbol{\omega}}, C'_{\boldsymbol{\omega}}$ are approximated as $\hat{C}_{\boldsymbol{\omega}}, \hat{C}'_{\boldsymbol{\omega}}$, computed using a Non-equispaced Fourier Transform (NFFT) algorithm given in [136]. The truncated Gaussian is a tensor product kernel. We compute the Fourier coefficients of a 1D Gaussian kernel of size n using MAPLE numerically. The Fourier coefficients of the truncated Gaussians are now approximated as the tensor product $\hat{G}_{\boldsymbol{\omega}}$. Hence, we approximate the scoring integral as $\sum_{\boldsymbol{\omega}\in I_n} \hat{G}^2_{\boldsymbol{\omega}}\hat{C}_{\boldsymbol{\omega}}\hat{C}'_{\boldsymbol{\omega}}e^{2\pi i\mathbf{x}.\boldsymbol{\omega}} = \sum_{\boldsymbol{\omega}\in I_n} \hat{F}_{\boldsymbol{\omega}}e^{2\pi i\mathbf{x}.\boldsymbol{\omega}}$.

Fourier coefficients of truncated kernels While shape complementarity involves Gaussian kernels, a different kernel is used in electrostatics. Following [135] algorithm computes approximations to these kernels using simple FFTs. For a kernel k, the Fourier coefficients $K_{\boldsymbol{\omega}}$ are approximated by the values of the FFT of k on a grid of size n^3. Our implementation follows the same algorithm, but the following technique can be used specifically for Gaussian kernels.

Fourier coefficients of truncated Gaussian The Gaussian function has a simple analytical expression for the Fourier transform and the truncated Gaussian is more involved. The error function $erf(z)$, the complementary

error function $erfc(z)$ are defined as $erfc(z) = 1 - erf(z) = \frac{2}{\sqrt{\pi}} \int_z^\infty e^{-t^2} dt$. Weideman in [170] provides a simple algorithm to compute the faddeeva function $w(z)$, which is related to the complex error function as follows: $w(z) = e^{-z^2} * erfc(-iz)$. His algorithm is elegant and given as 8 lines of MatLab code, which we have implemented. Let $a_1 + ib_1 = erfc(k\pi/w - iw/2), a_2 + ib_2 = erfc(n\pi/w + iw/2)$. Let $t = e^{-k^2\pi^2/w^2\sqrt{\pi}/(2w)}$. Now the k^{th} Fourier coefficient of Gaussian function with width w is $t(b_2 - b_1 + ia_1 - ia_2)$.

Fourier coefficients of weights of affinity functions The centers of the RBFs defining the affinity functions used in our scoring integral are not, in general, equally spaced on a grid lattice of reasonable sized. (Consider the atom centers, which are in R^3.) We use the technique of [136] to compute the Fourier coefficients of this function. Given M centers, located at general positions $\mathbf{x}_j$, with weights a_j, their NFFT' algorithm compute $\sum_{j=0}^{M-1} a_j e^{-2\pi i \mathbf{x}_j \cdot \mathbf{k}}, \mathbf{k} \in I_n$. Given sampling parameters α, m, the NFFT' algorithm computes the $n^3 = O(M)$ Fourier coefficients in $O(\alpha^3 M \log M + (2m+1)^3 M)$ time.

4.3.1.3 Inverse Peak Search

Using the previous algorithm, we are able to approximate the docking score $f(\mathbf{x})$ in terms of an inverse Fourier Transform. Depending on the formulation of the problem, we are either required to now find points where this function exceeds a certain threshold, or the positions where the top few function values occur. In the following, we assume that a certain threshold $\tau \in R$

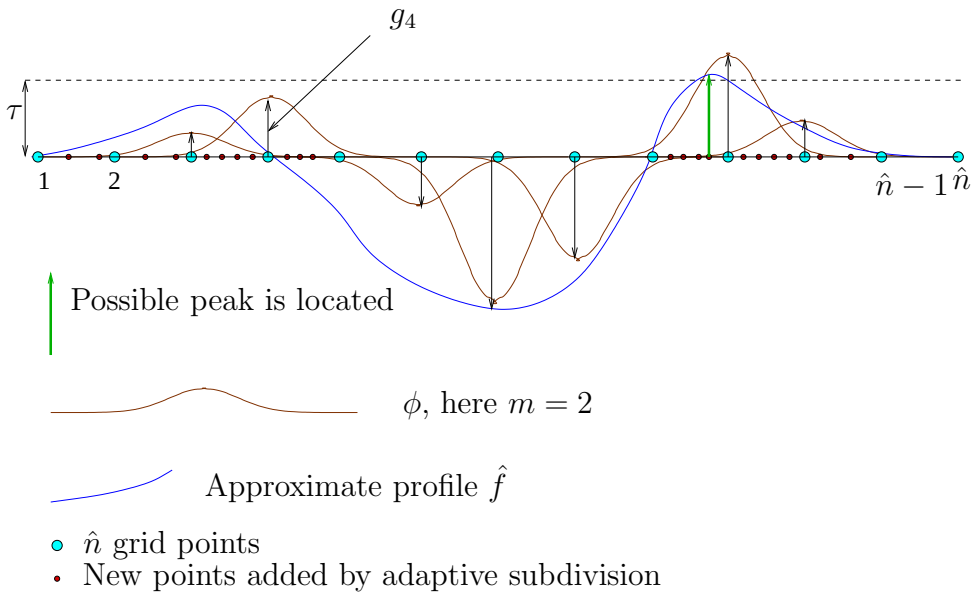


Figure 4.2: The docking peak search can be represented as finding the peak positions and values in a grid of overlapping splines.

is provided. Each interval in space, where this occurs is computed. The size of the interval is user defined. Formally, given the function $\hat{f}(\mathbf{x}) = \sum_{\boldsymbol{\omega} \in I_n} \hat{F}_{\boldsymbol{\omega}} e^{2\pi i \mathbf{x}.\boldsymbol{\omega}}$, we are required to compute $\{(\mathbf{x}, s) : s = Re(\hat{f}(\mathbf{x})) \geq \tau\}$.

Inverse FFT approach: A 3D IFFT of $\hat{F}_{\boldsymbol{\omega}}$ yields the docking profile $\hat{f}(\mathbf{x})$ at a uniform sampling. If we have prior knowledge on the smoothness of the profile, we can zero pad $\hat{F}_{\boldsymbol{\omega}}$ (if necessary) and obtain the profile at a sufficient sampling. This would generally lead to high computational and memory requirements. Instead, we first provide an approximation $\hat{g}(\mathbf{x})$ to the function $\hat{f}(\mathbf{x})$. Using the NFFT algorithm in [136], we make the following approximation:

$$\hat{f}(\mathbf{x}) \approx \hat{g}(\mathbf{x}) = \sum_{\mathbf{k} \in I_{\hat{n},m}(\boldsymbol{\omega}_\mathbf{j})} g_k \phi(\boldsymbol{\omega}_\mathbf{j} - \mathbf{k}/\hat{n}),\ (\mathbf{j} \in I_n,\ \hat{n} = \alpha n,\ \alpha \approx 2$$

Where a 3D grid of indices is represented by:

$$I_{\hat{n},m}(\boldsymbol{\omega}_\mathbf{j}) = \{\mathbf{j} \in I_{\hat{n}} : \hat{n}\boldsymbol{\omega}_\mathbf{j} - m \leq \mathbf{j} \leq \hat{n}\boldsymbol{\omega}_\mathbf{j} + m\})$$

Obtaining regions which are above a certain threshold is now reduced to finding roots of the previous polynomial. This is schematically represented in 1D in figure 4.2. In the figure, the blue line represents the function $\hat{g}(\mathbf{x})$ which we have computed as an approximation to $\hat{f}(\mathbf{x})$. The threshold τ is also marked, and we see that is one region where the function exceeds the threshold and is hence a possible docking site. There are $\hat{n}$ points in the grid, where $\hat{n} = \alpha n, \alpha \approx 2$. At each grid point k, we have a function ϕ, with weight g_k. As expected, core core overlaps provide low negative scores in the center, the fringes are 0 and in between lie the possible docking sites.

Solving roots of ϕ: If we use a cubic bspline function for ϕ with a support width of 5, it requires the root of a 7x7x7 system of degree 5 equations.

Adaptive search: We instead adaptively compute regions which satisfy our docking threshold using an adaptive search algorithm. We initially start with the $\hat{n}^3$ grid of ϕ as a set of intervals. We determine using a simple procedure if any interval can potentially contain a value greater than the docking threshold

and, if so, subdivide and recursively search the sub intervals. In figure 4.2, the new subdivisions are represented by red dots. Consider any interval I. There are multiple ϕ functions whose summation determine the function in I. If we change these ϕ, such that positive ones centered outside I come closer by one interval width, negative ones shift away from I by one interval width and positive ones centered inside I are given its maximum value, the sum of the new function at the interval endpoints defines an upper bound for the true function inside I. This gives us a criterion as to whether we need to further subdivide and check an interval or not.

Using a FFT for the 1st step: The docking profile is usually a thin closed surface with zeros on the outside and large negatives on the inside. Hence, in the very first step of the algorithm, a large number of regions are removed from further consideration. We are able to convert the algorithm in the first level into an FFT of size n^3. This is an efficient way of speeding up algorithm 1. We provide the analysis in 1D, which can be easily extended to 3D. Consider an interval $[i, i+1]$, with gaussian functions ϕ_k, where $i-m \leq k \leq i+1+m$, both positive and negative. Let the extent of the ϕ_k be m on each side of k. Let us construct a new function ψ_k by raising the value of ϕ_k to $max(\phi_k, \phi_{k+1}, \phi_{k-1})$ on the $\hat{n}^3$ grid. This gives us the following simple observation:

Lemma The summation of the ψ at a point k in the low resolution grid of the gaussian centers is always greater than the summation of ϕ at any point in any interval which includes k.

Algorithm 1 Inverse adaptive peak search

Inputs are:
    -$\hat{n}^3$: number of frequencies
    -h: accuracy of peak position
    -ϕ: Compactly supported smooth decaying function at each $k \in I_{\hat{n}}$
    -g_k: coefficients of ϕ
    -τ: threshold for docking score
    -$\{(val, pos)\}$: Current output peak regions and scores.
Preprocessing: [Interval set: $I = intervals(k)$]

while $I \neq \emptyset$ **do**
  $interval \leftarrow I.next()$

  if interval.isLowRes() **then**
    $t \leftarrow decisionFunction(interval)$
    if (t $> \tau$) **then**
      $I \leftarrow I \cup interval.subIntervals()$
    end if
  else
    $update(\{(val, pos)\}, interval)$
  end if

end while

Output: $[\{(val, pos)\}]$

Algorithm 2 Decision function

1: $\{\phi\} \leftarrow interval.overlapping\phi()$
2:
3: **for** $\phi \in \{\phi\}$ **do**
4: **if** $interval.isOutside(\phi)$ **then**
5: **if** $\phi > 0$ **then**
6: $t \leftarrow t + \phi(interval.cIdx(\phi.center))$
7: **else**
8: $t \leftarrow t - \phi(interval.fIdx(\phi.center))$
9: **end if**
10: **else**
11: **if** $\phi > 0$ **then**
12: $t \leftarrow t + \phi_{max}$
13: **else**
14: $t \leftarrow t - \phi(interval.fIdx(\phi.center))$
15: **end if**
16: **end if**
17: **end for**
18:
19: Output: $[t]$

The summation of functions ψ does not include any shifts. Hence, we can consider this as a convolution of ψ with g, the input to algorithm 1. Convolutions can be quickly computed in $O(n^3 \log n)$ using the FFT in a single step. This step eliminates most regions outside the overlap of molecules and core clashes from the docking profile. Hence, the adaptive search is limited to a narrow region where the surface contacts occur.

4.3.2 Rotational Search

If the active site of the proteins are not known, then a full 3D rotational search is employed. While the first protein is fixed, the second is rotated and a convolution based translation search is performed. For each such rotation, the current set of peak positions are updated as necessary. For a set of N_r rotations, the algorithm for the full docking search is as follows:

- **Preprocessing**
  - Set the current set of peaks to null: Peaks = {}
  - Scale the proteins such that their affinity functions fit inside $[-0.25..0.25)^3$
- For each rotation R, rotate the second protein and compute the translation docking score, and update the peaks
  - Let the second protein have M_2 centers convolved with a kernel K, with weights c_2.
  - $R(f_2) = \sum_{k=0}^{M_2} c_{2,k} R(K)$

- Call translational search with the two functions, the first protein's scoring function and the rotated function of the second.
- Update the list of peak positions if higher scores are received in any region.

- **Output:** Set of possible docking sites, along with the docking score.

4.4 Error and Complexity Analysis

To compute the overall error in our algorithm, we first analyze the approximations made in each step and provide bounds for each. Let us assume that a given affinity function is represented as the convolution between a kernel g and a set of coefficients c, c' for each molecule respectively. The error analysis is performed for the scoring using a given affinity and can be extended to multiple affinity functions as we just sum them up. As used previously, let $G_{\boldsymbol{\omega}}, C_{\boldsymbol{\omega}}, C'_{\boldsymbol{\omega}}$ be the Fourier coefficients of the kernel, and the weights used for affinity function of the two proteins. Let $\hat{G}_{\boldsymbol{\omega}}, \hat{C}_{\boldsymbol{\omega}}, \hat{C}'_{\boldsymbol{\omega}}$ be the respective approximations.

The exact docking profile is given as

$$\sum_{\boldsymbol{\omega} \in I_\infty} G^2_{\boldsymbol{\omega}} C_{\boldsymbol{\omega}} C'_{\boldsymbol{\omega}} e^{2\pi i \mathbf{x}.\boldsymbol{\omega}}$$

In the final step, where we compute the function using adaptive subdivision methods, we first convert this function into another, using the NFFT algorithm by Potts. This introduces an error ϵ_1 and we obtain $\sum_{\boldsymbol{\omega} \in I_n} \hat{G}^2_{\boldsymbol{\omega}} \hat{C}_{\boldsymbol{\omega}} \hat{C}'_{\boldsymbol{\omega}} e^{2\pi i \mathbf{x}.\boldsymbol{\omega}}$.

Given a user defined error ϵ, we derive the value of n such that

$$|| \sum_{\boldsymbol{\omega} \in I_\infty} G^2_{\boldsymbol{\omega}} C_{\boldsymbol{\omega}} C'_{\boldsymbol{\omega}} e^{2\pi i \boldsymbol{\omega}.\mathbf{x}} - \sum_{\boldsymbol{\omega} \in I_n} \hat{G}^2_{\boldsymbol{\omega}} \hat{C}_{\boldsymbol{\omega}} \hat{C}'_{\boldsymbol{\omega}} e^{2\pi i \boldsymbol{\omega}.\mathbf{x}} + \epsilon_1 ||_2 \leq \epsilon$$

.

We use the same analysis as our fast summation algorithm to obtain the errors in this algorithm. From our appendix A, lemma 1, the number of Fourier coefficients n required for a relative accuracy ϵ is:

$$n = min(\hat{n}) : \sum_{\boldsymbol{\omega} \in I_{\hat{n}}} G^4_{\boldsymbol{\omega}} \geq \frac{V}{2\pi} - \frac{M min_j(|(c^A c^B)_j|^2) V}{||c^A c^B||_1^2} (\frac{\epsilon}{3})^2$$

$$V = \int g^2 \; in \; (-0.5..0.5]^3, \; M = max(M_1, M_2)$$

The above equation gives us a practical method to compute n, although we use a binary search to get a tighter bound in our implementation. We are further interested in obtaining a theoretical bound for n given a certain accuracy requirement to see how it grows with the number of atoms in the system, or in the general case of affinity functions, to see how it grows with the number of RBF centers defining the affinity function.

Size and resolution vs. cost: The size of the molecule and the resolution we are interested affects the cost of the algorithm. The resolution parameter can affect the rate of decay of the Gaussian (b) and the number of centers being

considered. From the results presented in appendix A, we see that n^3 varies as $M(\sqrt{b/2})^3/\epsilon^2$, where M is the maximum number of centers being used in scoring given the affinity functions.

Cost of the algorithm: The Fourier coefficients of the truncated Gaussian $G_{\boldsymbol{\omega}}$ are precomputed to full precision using MAPLE. The NFFT algorithm has been implemented to compute $C_{\boldsymbol{\omega}}, C'_{\boldsymbol{\omega}}$, the Fourier coefficients of the sum of centers in $O(M_1 + n\log n), O(M_2 + n\log n)$, respectively, where n, computed using our error analysis, is shown to be $O(max(M_1^{1/3}, M_2^{1/3}))$. Computation of the product of the coefficients is costs $O(n^3)$. An IFFT of the product yields the docking profile on a n^3 resolution grid and costs $n^3 \log n$. To obtain the peaks in a higher resolution, without using a larger grid, we can perform the inverse peak search algorithm described in §4.3.1.3. If there are η regions which satisfy the threshold τ in the docking profile, they can be located and computed in a grid of size 2^h in $O(\eta h n)$. Hence, the computational cost of our docking algorithm grows linearly with the number of atoms in the molecules. Since each rotation is performed independently, the total cost is $O(max(M_1, M_2)N_R^3)$, where N_R^3 is the number of sampled rotations. The memory cost is $O(max(M_1, M_2))$. Compared to traditional grid based algorithms, we see that our algorithm has lower computational costs and lower memory requirements according to analysis by Dr Castrillon in [30]:

	Time complexity	Space complexity
Ritchie et al	$O(D_\rho N^7)$	$O(D_\rho N^3)$
Kovacs et al	$O(D_r D_\rho N^7 \log N)$	$O(D_r N^5)$
FFT	$O(N_r^3 N^3 \log N)$	$O(N^3)$
Our method	$O(N_r^3 M \log M)$	$O(M)$

where M represents the maximum number of centers defining the RBF affinity function, N_r^3 are the number of orientations considered, N^3 for the FFT technique is the size of the grid, and the rest are sampling parameters as detailed in [29].

4.5 Flexible Docking

Docking involves a change in shape of either or both proteins, and especially the smaller or ligand molecule. The changes in shape occur either due to backbone movements or side chain movements. Movement of loops and domains leading to large conformational changes occur due to backbone torsional angle changes. Rearrangement of side chains in binding regions occur due to torsional angle changes in the side chains of various residues. Figure 4.3 shows three residues along a typical backbone with different relevant torsional angles marked. The backbones motion is mainly controlled by the pair of ϕ, ψ angles given for each residue. The side chains move through torsional changes in the χ angles. Depending on the amino acid type, there can be up to 5 such successive angles.

Computing torsion angles Torsion angles are defined using 4 points. Let the first three points define plane p_1 and the last three plane p_2. The angle between the normals of the two planes, given by the inverse cosine of the dot product is the torsion angle of the bond between the second and third point.

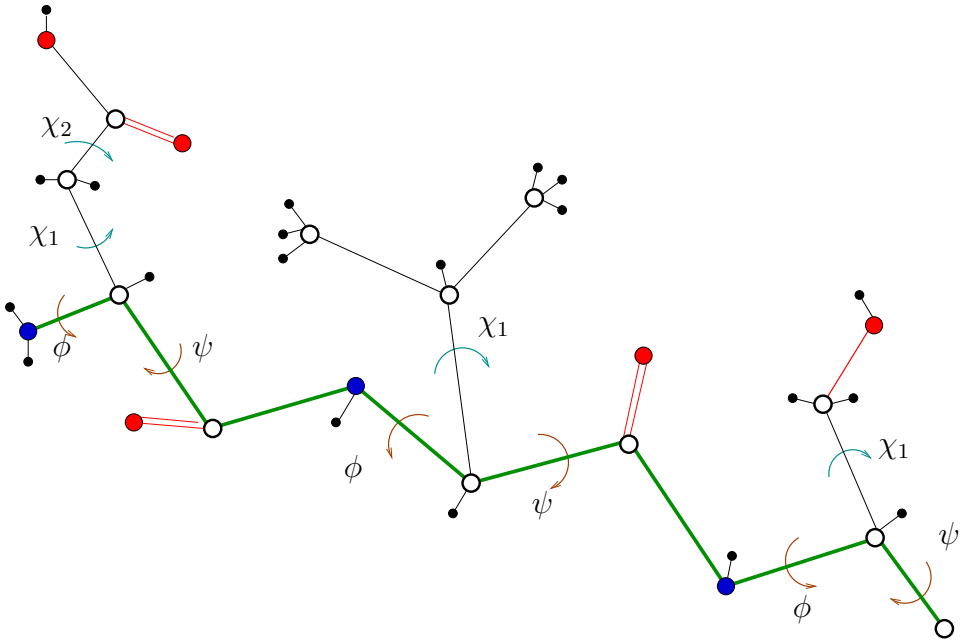


Figure 4.3: The first three residues of 1AY7.pdb: ASP, VAL, SER are shown schematically with the relevant backbone (ϕ, ψ) and side chain (χ_1, χ_2)torsion angles.

4.5.1 Flexibility in proteins

Flexibility analysis of proteins can be performed through a wide variety of algorithms.

- **Molecular dynamics** Molecular dynamics involves simulation of the protein in a solvent environment and saving the conformation state at regular time intervals. Since this simulation is often at very small time scales, (pico or nano seconds), large conformational changes (which occur over micro or milli seconds) will not be recorded. Hence obtaining flexibility analysis through molecular dynamics is limited. An adaptive solver is given in [90]. By allowing users to interact with the system, conformational changes can be forced and observed [107], [160]. A multiple grid method for solving the electrostatics efficiently [156]. Compact structural domains were computed in [76] using simple force calculations in a protein structure.

- **Xray Crystallography and Nuclear Magnetic Resonance (NMR)** Xray Crystallography is used to obtain high resolution images of proteins, upto the atomic level. Most structure in the PDB are generated using this method. NMR techniques have been used to obtain dynamic conformations of proteins. Given the large number of states which could be obtained from molecular dynamics, NMR and xray crystallography, the following methods generate certain important conformal states by reducing the number of degrees of freedom in the protein.

- **Comparison of conformal states** Protein dynamics give rise to a large number of conformations. Analyzing these conformations for any problem, including flexible protein docking is not computationally feasible. Hence many methods are used to reduce these conformations to a new basis, where the principal basis gave the large fluctuations efficiently. Conformational changes of a protein is shown to be captured by using only a few bases and projection vectors (See [163] and [162]). Normal mode analysis and principal component analysis are two methods to reduce the dimensionality of the problem. Singular Value Decomposition (SVD) is commonly used to find basis vectors to reduce the dimensionality of a set of vectors. An equivalent formulation using Principal Component Analysis (PCA) is also done. Consider the column vectors of a matrix A as the zero mean weighted atomic displacement positions. In [65], a theorem relating the atom displacements to the frequencies of vibrations is presented. In this paper, the authors prove that if a large molecule only flexes around a certain minimal energy state that is approximated by a multidimensional parabola, then the average displacements of the atom positions is the sum of the contributions from each normal mode, which is proportional to the inverse square of the frequency. For Normal Mode Analysis (NMA), the moment matrix diagonalized is $A = k_B T F^{-1}$, where k_B is the Boltzmann constant, T the absolute temperature and F a matrix of the second derivatives of the potential energy at a minimum point. Successful modeling of

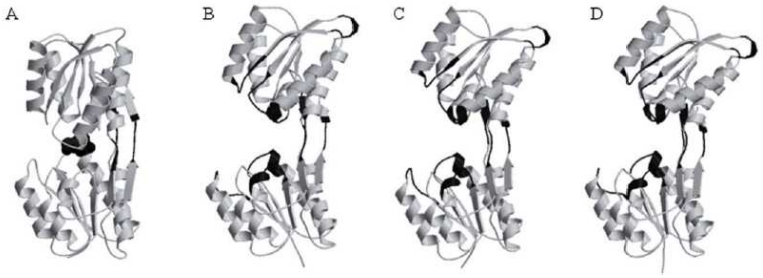


Figure 4.4: ALBP hinge bending, image from [114]

the Chaperonin GroEL was performed using NMA in [113]. To avoid the computations on a large matrix, [161] compute a blocked version of NMA by grouping residues. Gaussian Network Models (GNM), that use Kirchoff matrices instead of the true Hessian, were introduced for proteins in [92].

- **Side Chain Flexibility**

  Clustering the various known conformations for a given residue shows that only a few preferred states are present predominantly in nature. Once high resolution structure information for proteins were available, Ponder and Richards showed that the variation in the cluster was much lower than previously thought [134]. Both backbone independent and backbone dependent libraries are given by Dunbrack-Jr and Cohen [48]. The Dead End Elimination (DEE) theorem from Desmet in [45] is often used to prune the number of rotamers to consider for a global energy minima. Let energy be computed as: $E_{global} = E_{template} + \sum_i E(i_r) + \sum_i \sum_j E(i_r j_s), i < j$, where E_{global} is the total energy in the protein that

we want to minimize, broken up as the energy due to the backbone structure and properties: $E_{template}$, the sum of the interactions of residues with the backbone: $\sum_i E(i_r)$ and pairwise interaction energies between residues. Given two rotamers i_r, i_t, the theorem states that if $E(i_r) + \sum_j min_s E(i_r j_s) > E(i_t) + \sum_j max_s E(i_t j_s), i \neq j$ then i_r does not belong to the global minima.

4.5.1.1 Protein Domain Analysis

To construct a flexibility model, we first need to identify rigid domains in a protein. There have been a variety of techniques which can compute rigid domains from one or more conformation of a protein.

Non polar regions in protein tend to lie in the interior and this hydrophobic effect folds the protein. In [181], the authors describe how to capture this information into rigid domains of the protein. Their assumption is that rigid domains folded by the hydrophobic effect behave as a *compact unit* during conformational changes. To quantify this, they hierarchically grouped residues in a protein to form a tree, using a coefficient of compactness Z given by the ratio of the accessible surface area of a domain to the surface area of sphere of equal volume. Gerstein, Lesk and Chothia analyzed a large number of known protein movements and classified them into collections of hinge and shearing movements [64]. Difference distance matrices and a full search is used in [128]. Static core or the backbone of molecules and their associated rigid domains were computed in [23] using two different conformations of a given protein. α

helices, β strands and loops were segmented. Similar pairs of segments were clustered in a tree-like fashion using a rmsd calculation. Domains or compact units of a protein were also computed by [155]. The heuristic they used was that the amount of internal contact a domain had was larger than the amount of contact it had with the rest of the protein. Hence by choosing suitable split planes along the sequence, they form compact sequences. Extending this idea, a Monte Carlo sampling in internal coordinates using relevant torsion angles was performed in [115]. They obtained a set of low energy conformations for any given protein structure as a representation of its flexibility.

Some of the software to compute domains in proteins are HingeFind [175], DynDom [74] FIRST [81] and DomainFinder [75]. FIRST requires a single conformation while HingeFind and DynDom require two. DomainFinder uses a single conformation also, but is also capable of using two conformations for domain analysis. DynDom computes the domains using 2 conformations. In particular, they use one conformation and a set of displacements to form another conformation, from either a second crystal structure, dynamics or NMA. Rotation vectors are computed using a sliding window for main chain segments. K-means clustering is employed to compute clusters of rotation vectors. Then residues are classified into domains or interdomain residues. The transformation is described as a general screw twisting motion and hinge axis are identified and also further classified as closure and twist axes. HingeFind again uses two conformations of the protein, obtained as different crystal structures or dynamics computations. They select subsets of C^{α} atoms around

randomly chosen points from both conformations and compare for domains. Every atoms which show large movement in the two subsets are removed from the domain and neighbors are screened and added if required. Then the atoms in the newly found domain are deleted from both conformations and the procedure is repeated at the next randomly chosen atom. Since they are interested in obtaining hinges, the 6D transformation is approximated as a rotation. Using graph theoretical algorithms, [81] obtain flexible and rigid domains from a single conformation of a protein. Their software FIRST (Floppy Inclusion and Rigid Substructure Topography) computes redundant constraints between atoms, by checking it through the absence of changes in zero eigenvalues of the dynamical matrix before and after addition of the constraints. It then checks for contiguous sets of tetrahedrons in the network marks them as domains. Since this procedure is $O(N^5)$, they use an order $O(N^2)$ integer algorithm using ideas from the 3D pebble game [80]. We use DomainFinder from Hinsen to compute our flexibility model. It computes NMA on the input protein and allows the user to select a set of modes and a deformation threshold to compute rigid domains. Cubes of approximately 6 residues are given 6 transformations from the selected modes. Clusters of cubes with similar transformations are grouped to form a rigid domain.

4.5.2 Labeled Flexible Chain Complex

The geometry of the Flexible Chain Complex (FCC) is introduced in 3.1.3. Here we label the FCC with flexibility information to provide conforma-

tional sampling for docking. Movement of loops and domains leading to large conformational changes occur due to backbone torsional angle changes and hinge type bending and shearing movements. Rearrangement of side chains in binding regions occur due to torsional angle changes in the side chains of various residues. In section §4.5.1, we have already summarized various algorithms to compute flexibility in proteins. Here we compute a hierarchical decomposition using Normal Mode Analysis. But since our data structure is general, we observe that users can augment their own descriptions to it. The FlexTree is a data structure introduced by Zhao, Stoffler and Sanner [182], and provides a method for storing a hierarchical description of flexibility. Their data structure should be easily parsed into our Flexible Chain Complex and used in the flexible docking algorithm. Our data structure provides similar capabilities, but with shear, bending and twist motions and we maintain a general graph with priorities at each edge and provide an automated algorithm to compute the flexibility. We first describe the flexible data structure and then provide an automated algorithm to compute it for any given protein.

4.5.3 Hierarchical Domain Identification

Domains are considered to be rigid contiguous parts of a protein which show little movement in different conformations (obtained through molecular dynamics, normal mode analysis etc.). Given a threshold of rigidity, a protein decomposes in to different number of domains. Hence, we can automatically obtain, from dynamics or other simulations, a hierarchical domain decompo-

sition for proteins. In particular, we keep a decomposition tree, with each discrete level representing the protein at a rigidity threshold. Since we provide a output ASCII file format, users can update any kind of flexibility at desired locations, with ranges. Below, we provide one method to automatically compute all the required quantities.

Connectors, Flexible Loops and Domain Model At any given level, there are various domains which interact either through connected chain segments or large interfaces. In particular, we call these chain segments and areas as connectors. Domains and connectors form a complete description of the flexible protein at a given level. We also recognize that domains have flexible loops and chain ends on their surfaces. We identify and mark these are flexible loops, apart from the rigid segments in a domain. To build a formal model, we define the following objects:

- **Segment** A segment is a contiguous sequence of residues from the chain of a protein.
- **Flexible loops** A segment on a rigid domain surface, not acting as a connector to another domain and also consists of large flexible side chains. It is useful to identify such flexible regions in a domain to provide a finer resolution flexibility model for the docking algorithm.
- **Domain** A connected set of rigid segments and flexible loops. Using different rigidity thresholds, we obtain a hierarchy of sub-domains.

- **Connector** Segment between two domains. We also consider large domain interfaces as connectors.

- **Flexor** A set of connectors between a pair of domains, associated with certain flexibilities. The flexors are given priorities over all levels, to form a hierarchical description of protein flexibility. Unlike in the FlexTree, Flexors here need not form a cut (removal of the connectors in the flexor need not divide the component into two).

4.5.3.1 Motions Allowed at Flexors

We provide shear, bending and twist motions at flexors. In [115] techniques to compute such motions for two domains linked by a single or double stranded linkers is provided.

Shear This describes lateral movement along interfaces between domains. The magnitude of shear is limited by a maximum value chosen by the user and the length of the smallest connector between the two domains under consideration. In the absence of connectors, the line joining the centroids of the two domains is used to compute the normal of the shearing plane.

Primary and Secondary Bending Hinge motions are represented by two perpendicular rotations at a hinge point and primary, secondary axes. Such motions are given only between domains which have a definite connector between them. Domains which share a large interface area but no connectors are

only provided with shearing motion. Consider k connectors $c_1, ..c_k$, with end points $\mathbf{e}_1^1, \mathbf{e}_1^2, ..\mathbf{e}_k^1, \mathbf{e}_k^2$. Let the mid points of each set of end points be $\mathbf{c}^1, \mathbf{c}^2$. We choose the connector c_{hinge}whose ends are closest to these averages and choose its center as the center of rotation. The center atoms position of c_{hinge} is chosen as the hinge point. The normal of the plane containing its end points and mid point is used as the hinge axis. The perpendicular to this is used as the secondary hinge axis. These values are not fixed to the initial structure, but updated with each new conformational sampling, before obtaining a new one.

Twist When a single physical connector exists between two domains, it is also given a twist motion by updating torsion angles along its backbone.

4.5.3.2 Normal Mode Analysis

Normal Mode Analysis for a given unbound structure of a protein is computed using Hinsen's DomainFinder program [75]. For a given deformation threshold and domain coarseness, a set of rigid domains with their similarity indices is computed. Their output defines the domains as a set of contiguous residues. These are collected as segments of the protein. Let $S_1, .., S_d$ be the set of segments in the d domains at a given level. By deleting these sets from the protein, we are left with segments which form either flexible loops, connectors between domains or ends of chains. We assume that a chain consists of atleast one domain. Virtual connectors are added to domains which share a common

interface. If we are dealing with a large macromolecule, more than one level can be computed by varying the parameters to DomainFinder.

4.5.4 Flexible Docking Algorithm

The flexible docking algorithm consists of adaptively sampling conformation space. In the first step, the high priority flexors are used to compute a set of conformations, the size of which is given by the user. A low resolution representation of the proteins are used to compute docking at each of these conformations over all of orientation space (or limited by the binding sites if known). Given a set of possible docking positions, the domain(s) in that regions is further subdivided and a new set of conformations are computed for docking. If the sub domains (whose union is not the parent domain due to the presence of flexible loops) are far away from the interaction region, then only conformation sampling of the flexible loops are considered. In the last step, we refit all the interface side chains using a greedy algorithm.

Multiresolution Sum-of-Gaussians Representation The electron density of an atom at a point $\mathbf{x}$ is represented as a Gaussian function: $f(\mathbf{x}) = e^{\beta(\frac{|\mathbf{x}-\mathbf{c}|^2}{r^2}-1)}$ where $\mathbf{c}, r$ are the center and radius of the atom. and thus the electron density of a protein with M atoms at $\mathbf{x}$ is: $f(\mathbf{x}) = \sum_{i=0}^{M-1} e^{\beta(\frac{|\mathbf{x}-\mathbf{c}_i|^2}{r_i^2}-1)}$ where β is a parameter used to control the rate of decay of the Gaussian and known as the *blobbiness* of the Gaussian (see section §3.3). By clustering atoms and varying the rate of decay of clustered Gaussians, we can obtain a

Sum-of-Gaussians representation for the protein at multiple resolution levels.

Soft Docking We use the algorithm described before in section §4.3 for soft protein-protein docking. Given two conformations, the algorithm predicts a set of possible docking sites where the docking score is above a user defined threshold. In particular, given N scoring functions $f_{1,k}, f_{2,k}, k = 1..N$and a user defined score τ, we solve the equation:

$$\{(\mathbf{t}, \mathbf{r}, s) : (s = Re(\sum_{k=1}^{N}(\int_{\mathbf{x}} f_{1,k,\beta}(\mathbf{x}) T_{\mathbf{t}}(\Delta_{\mathbf{r}}(f_{2,k,\beta}(\mathbf{x})))d\mathbf{x}))) \geq \tau \forall (\mathbf{t}, \mathbf{r})\}$$

In the above equation, $\mathbf{t}, \mathbf{r}$ is the translation and rotation space we require to sample. The resolution of the maps is controlled by β. In particular, our soft docking algorithm can be restricted to the orientations we are interested in and the resolution of docking maps can be controlled.

4.5.4.1 Hierarchical Docking

Our flexible docking has three stages: Parallel docking of a global hierarchical conformational sampling, local flexible loop and side chain sampling and interface fit using a greedy algorithm.

FCC Construction:

1. ***Input:*** For a given protein, Normal Mode Analysis is used to compute, for L levels, the domains D_i for each level.

(a) $D_i = \bigcup S_{i,k}$, a set of k segments.

2. ***Compute flexible loops and Connectors:*** Follow each segment $s \in D_i$. If it terminates back in D_i without crossing into any other domain, or is the end of a chain, add it to D_i as a flexible segment f.

   (a) $D_i = D_i \bigcup F$, a set of flexible segments.

   (b) Any segment c that crosses from D_i to $D_j, i \neq j$ is added to a list of connectors: $C = C \bigcup c_{i,j}$.

3. ***Compute Labeled Flexors:*** For each pair of domains D_i, D_j, all c_{ij} are collected to form a Flexor between the domains. If $\{c_{i,j}\} = \Phi$, then the area of the interface is used to determine if we need a virtual flexor or not.

4. ***Compute Hierarchy:*** Steps **2**, **3** are repeated for all levels. Domains are broken up if necessary to maintain unique parent domain nodes.

5. ***Output:*** This labeled complex is printed out in a easy-to-use ASCII file. Users can intuitively add/delete new domains, connectors and flexors at will.

Adaptive Sampling of Conformations The biased probability Monte Carlo sampling in [115] can be applied to our structure, but here we provide a random sampling followed by a steric collision test. There are two distinct types of flexors: Flexors which lead to a cut in the component and those which

do not. For each flexor, we arbitrarily assign a left and right domain, and always update the right domain. For a flexor which defines a cut, all domains to the right are updated, while for the other case, only the right domain is updated. The connectors from the right to other domains are updated to maintain structural integrity of the protein. This reduces the range of motion at a flexor which is not a cut. Each flexor is given a score depending on the range of motions computed for its associated shear, primary/secondary bending and twist. The sampling is adaptively performed to reflect these scores.

Global Conformation Sampling and Low Resolution Search:

1. ***Input:*** The FCC of a ligand and a fixed number of global conformations N.

2. ***Allocate Conformations:*** Given the set of Flexors $\{f\}$ at the top level, a hierarchy of importance is built and the total number of conformations is divided among them.

   (a) Determine if each Flexor f is a cut or not. The domains, connectors at any level form a graph. Hence, each flexor, defined as the set of connectors between domains can possibly disconnect the graph.

   (b) If f is a cut, then let $\{d_l\}, \{d_r\}$ be the set of domains to the left and right. Let the sum of their weights be w_l, w_r. Then the score for f is $s_f = min(w_l, w_r)$.

(c) If f is not a cut, let w_{dl}, w_{dr} be the weight of the left and right domain. Then the score for f is given as $s_f = min(w_{dl}, w_{dr})$.

(d) N_f, the number of conformations allocated to f is N^{s_f/Σ_f}.

(e) Each flexor is associated with at most 4 motions: Shear, bend, secondary bend and twist. We use heuristics based on their computed range of motion to assign sampling for each.

3. ***Compute Conformations:*** We recursively apply a new motion at each flexor.

   GetConformation (Flexor f_i, Molecule m)

   - Set $\hat{m} \leftarrow m$

   - For each motion t in f

   - apply($f, t, \hat{m}$)

   - If (i = N), Print($\hat{m}$)

   - Else Call GetConformation($f_{i+1}, \hat{m}$)

   - End for

   Output: A set of at most N conformations of the ligand, with higher priority flexors given a higher resolution sampling.

4. ***Low Resolution Search:*** A 20 degree rotational sampling is used for soft docking. We use residue level parameters to represent the shape affinity functions.

- Electron density can be represented as a sum of Gaussians, $\sum e^{-\beta(x-ci)^2/ri^2}$ as described before. The FCC gives a clustering of atoms into residues. For this lower resolution search, we can either decrease the decay parameter, or use fewer Gaussians to represent a residue.
- For every conformation $\hat{m}_i, i = 1..N$, call F^2Dock.

5. ***Output:*** Conformations and their orientations where the docking score exceeded a user defined threshold.

Finer Resolution Search Given a set of promising orientations from the previous low resolution search, soft docking is again performed with high resolution affinity functions. For shape, we vary the rate of decay of Gaussians representing atoms to obtain a higher resolution map. We use a value of -2.3 to represent the atomic level resolution. For electrostatics, we use a charge from OPLS at each atom. Each conformation and orientation saved from the low resolution search is used as input. Hence, we adaptively sample orientation space and use a multiresolution representation of affinity functions. To further improve the docking score, we perform a refitting of side chains at potential interfaces.

Refitting Side Chains at Interfaces We use the Dunbrack [48] backbone independent library to sample interface rotamers. In table 4.1, the number of rotamer angles and rotamer conformations in the library is given.

Residue	Num R	Num χ	Residue	Num R	Num χ
ARG	81	4	LYS	81	4
ASN	18	2	MET	27	3
ASP	9	2	PHE	6	2
CYS	3	1	PRO	2	2
GLN	36	3	SER	3	1
GLU	27	3	THR	3	1
HIS	9	2	TRP	9	2
ILE	9	2	TYR	6	2
LEU	9	2	VAL	3	1

Table 4.1: The number of rotamers (Num R) and the number of torsion angles along the side chain for each type of residue from the Dunbrack backbone independent library is summarized.

Given a certain conformation from soft docking, we would like to optimize the side chains in the interface to obtain a better fit of the proteins. Let there be N interface residues in the given conformation, and the residues be $R_i, i = 1..N$. Each residue is associated with a rotamer set $\{r^i\}$. The cardinality of this set depends on the type of the residue. Since we do not want to discard the current conformation for a given side chain, we include R_i into the set $\{r^i\}$. From the Dunbrack library, we also have $\{p^i\}$, probabilities for each rotamer for a given side chain. We set a probability for the current side chain as equal to the highest in its set of rotamers. Let the scoring function for the j^{th} rotamer for side chain i be $S(r^i_j)$. A solution is any set $\{r^i_j, i = 1..N\}$, and is the optimal solution when $\sum_i S(r^i_j)$ is maximized.

Intersection lists

Using the FCC of the second protein, residue intersections are computed for all surface residues. Surface residues are computed as those whose atoms (at least 1) are less than 4Åaway from the other protein's surface. Given a residue, we traverse down the FCC hierarchy to adaptively compute all other intersecting residues. Let $\{I_i\}$ be the set of residues which intersect residue R_i and any of its rotamers. Assuming a maximum number of intersecting residues N_I, the cost of this algorithm is linear in the number of residues. Here, we are interested in computing intersection with neighbors, assuming that the current residue can be any of its rotamers.

Scoring

The addition of a residues rotamer into the current partially formed interface will influence both the current score of the docking and the scores of potential rotamers yet to be added. The score is currently computed as the sum of shape complementarity and electrostatics. First we compute the function ϕ_s, ϕ_e, density and electrostatics for the first protein. For any given atom in a rotamer under consideration, we calculate the approximate Lennard Jones score depending on its distance from the electron density, and the electrostatics score as a product of its negative charge and the field ϕ_e. A slightly higher rating is used for the electrostatics energy contribution. The addition of a new rotamer affect other rotamers yet to be added. The scores of those rotamers are updated using a simpler scheme based on steric overlaps with the newly added rotamer, to avoid costs of recomputing the fields.

Side chain repacking algorithm:

From the results of the previous docking steps, we are given a protein and ligand (in possibly new conformations) and a transformation between the proteins that yields a good docking score. Now we proceed to repack the side chains of the ligand to improve the fit. First, we compute the interface residues for the given transformation. This can be quickly done by a pre-computation of the signed distance function of the individual proteins. Next we compute all the rotamers of all the interface residues, by looking up appropriate entries in a Rotamer library. In the third preprocessing step, we compute all neighbors of a given residue. We can assume this to be a constant number. In the final step, we compute the current score for each residue and its set of rotamers. The score is currently a sum of both shape and electrostatic interactions.

Now we reinsert a new rotamer in place of each of the original interface residues. The choice is a greedy choice, with a backtracking option when the score is lower than a threshold. Hence, we first insert the rotamer with the highest docking score among all interface residues and all their rotamers. The potential costs of all neighboring residues and rotamers can now be updated (currently we use a simple steric test for performance reasons). In this greedy fashion, all interface residues are replaced by new rotamers. Since the current position of a residue may be its most stable state, we also include the current residue as one of its choice of rotamers. If the total score at any point is too unfavorable, we backtrack and use the probabilities in the Dunbrack rotamer library to pick a new choice.

Assuming that each residue has a fixed number of rotamers and a fixed number of neighbors, the cost of maintaining a dynamic list of scores for residues for N interface residues and their insertion into the ligand should be $O(N \log N)$, but our current implementation uses a simpler $O(N^2)$ update. The main steps in the algorithm is presented below.

1. ***Input:*** The FCCs of proteins A and B, with a list of transformations from soft docking, $\{X\}$, that lead to potential complexes.

2. ***Output:*** For each $X \in \{X\}$, multiple sets of new repacking of side chains at interface.

3. ***Preprocessing:***

   (a) Potential fields ϕ_s, ϕ_e.

   (b) Interface residues $\{R_i, i = 1..N\}$ of protein B.

   (c) Rotamer set $\{Rot_{i,j}\}, i = 1..N, j = 1..n(R_i)$ and probabilities $p_{i,j}$.

   (d) Neighboring residues $\{R_i^N\}$ for every interface residue R_i.

4. ***Fit interface for each X:***

   (a) Transform second protein:

   $$Trans(B, X)$$

   (b) Compute relevant interface residues $R_i^X, i = 1..N_X$ from R_i.

(c) Compute initial scores for rotamers: $\{S(Rot^X_{i,j})\}$.

(d) ***Incremental greedy fit:*** Repeat till we get a desired number of (suboptimal) solutions within finite number of attempts.

i. Initialize $Sol = \Phi$.

ii. Choose next best fit: $R_b = argmax_{Rot_{i,j}}\{S(Rot_{i,j})\}$, $Sol = Sol \cup R_b$.

iii. Update scores:

$$S(R_b) - = StericScore(R_b, R^N_b).$$

iv. Discard $\{Rot_i\}$.

v. Feasibility test: If $S(Sol) < \tau$ discard Sol. Else, continue with step ii.

(e) ***Evaluate:*** For each solution, compute and print RMSD with true solution.

4.6 Summary

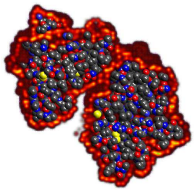
Our algorithms are based on representing affinity functions in a multiresolution radial basis function format. The smoothed particle representation, together with non-equispaced Fast Fourier transforms allows us to design and analyze our algorithm without the use of a sampling grid. The soft protein-protein docking algorithm is built upon accurate construction of molecular surfaces and properties. Its efficiency and multiresolution nature is

utilized to sample conformational space and allow flexible docking. A simple greedy hueristic based algorithm is also presented to improve potential docking interfaces.

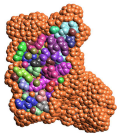
Chapter 5

Docking Results

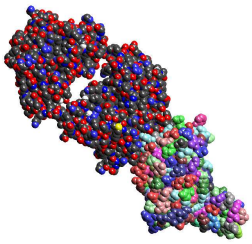
We have computed the docking predictions for a set of 71 complexes using different affinity functions and flexible models. We have also taken a set of three test cases to compare, at each step, with a traditional grid based approach. The three complexes we use to compare are: Hyhel-5 fab complexed with bobwhite quail lysozyme (1BQL.pdb), Idiotype-anti-idiotype fab complex (1IAI.pdb) and an influenza virus hemagglutinin complexed with a neutralizing antibody (2VIR.pdb). We will simply refer to these complexes as **complex 1, complex 2, complex 3** respectively (see figure 5.1, where the first molecule is colored using standard atom colors while the atoms in the second molecule are colored by their residue type to differentiate the two molecules in the complex. The three molecules/skins in the first column had 3263/4519, 3342/4555, 3243/4308 atoms/kernel centers respectively. In the second column, there were 988/1087, 4956/1719 and 5293/469 surface and interior atoms respectively.).



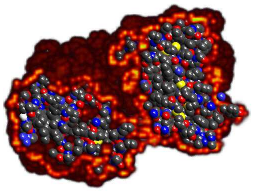
(a) An immunoglobulin from the influenza virus (IGG1, LAMBDA) (VIR.pdb)



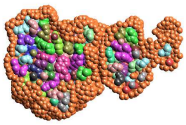
(b) Part of hemagglutinin (2VIR.pdb)



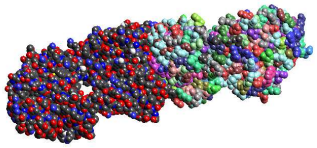
(c) **Complex 3:**influenza virus hemagglutinin complexed with a neutralizing antibody (2VIR.pdb)



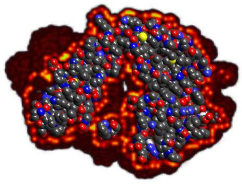
(d) Idiotypic FAB 730.1.4 (IGG1) of virus neutralizing antibody(1IAI.pdb)



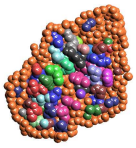
(e) Anti-Idiotypic Fab 409.5.3 (Igg2A)(1IAI.pdb)



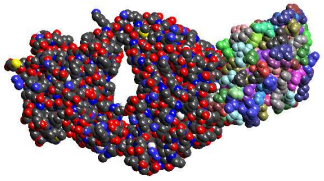
(f) **Complex 2:**Idiotype-anti-idiotype fab complex(1IAI.pdb)



(g) Hyhel-5 FAB (1BQL.pdb)



(h) Bobwhite Quail Lysozyme (1BQL.pdb)



(i) **Complex 1:**Hyhel-5 fab complexed with bobwhite quail lysozyme (1BQL.pdb)

Figure 5.1: The three complexes we have used as test cases. In the first column, we show one protein of the complex with the grown surface in red. The second column shows the surface atoms in light brown. We show a cut away to reveal the two skins. In the last column, the complexed structures are shown.

5.1 Soft Docking

For soft docking, we first use shape complementary as the only affinity function in scoring. Then we investigate the effects of introducing electrostatics interactions.

5.1.1 Comparison with Grid-based FFT Algorithm

Difference in docking profiles: We compared the difference in the energy of the docking profile we obtain to that obtained from a 256^3 grid. From tables 5.1,5.2 and 5.3, we see that the differences are very small for relatively fewer Fourier coefficients. A slice from the docking profiles the two methods are shown in figure 5.2. From the figure, we can see that the shape of the profile and the location of the peaks are well conserved in our algorithm.

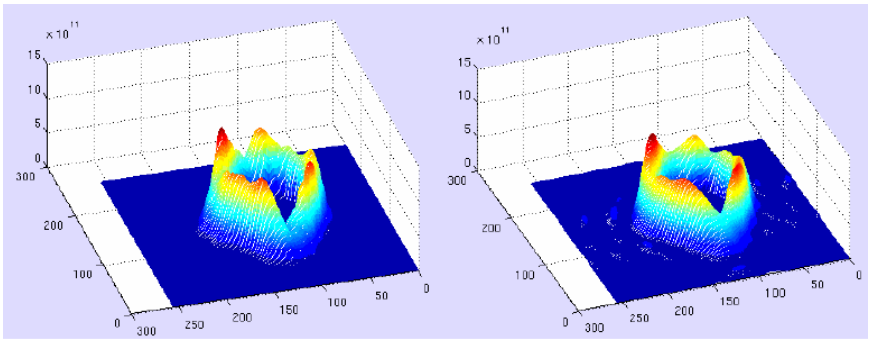


Figure 5.2: Comparison of a slice from our docking profile compared with that of a FFT based algorithm on a 256^3 grid. The shape and location of peaks is shown to be well conserved.

Number of freq.	$\beta = -0.5$		$\beta = -1$	
	l^2	l^∞	l^2	l^∞
16^3	6.3364	3.0409	9.9454	3.5909
20^3	3.9761	1.2994	7.9016	1.7434
32^3	1.1991	0.2889	5.3285	0.5909

Table 5.1: Difference in energy, in %, for **complex 1**, with $\alpha = m = 2$ as the NFFT parameters

Number of freq.	$\beta = -0.5$		$\beta = -1$	
	l^2	l^∞	l^2	l^∞
16^3	4.5203	3.5743	6.8897	4.2208
20^3	2.5131	1.4592	5.1096	1.8793
32^3	0.8462	0.2480	3.6941	0.5297

Table 5.2: Difference in energy, in %, for **complex 2**, with $\alpha = m = 2$ as the NFFT parameters

Number of freq.	$\beta = -0.5$		$\beta = -1$	
	l^2	l^∞	l^2	l^∞
16^3	4.8228	2.0457	7.7806	2.3983
20^3	2.7570	0.8029	6.0601	1.0721
32^3	0.9504	0.2017	4.6343	0.4111

Table 5.3: Difference in energy, in %, for **complex 3**, with $\alpha = m = 2$ as the NFFT parameters

Comparison with FFT grid-based algorithm using larger set of complexes: We compare results for redocking 71 complexes using shape complementarity to a traditional, expensive 128^3 grid FFT based docking. We find where the true position lies in our ranking of peaks. We use an accuracy of 2 Åbetween what we have and the true docked complex position while searching for the peaks. We see that the best results are obtained with a rate of decay around the recommended value of -2.3. These results have been plotted for comparison in figures 5.3, 5.4 and 5.5. Our new algorithm uses far lesser time and memory than the FFT grid-based method. In this experiment, we used Euler angles as they are used by many groups who perform the FFT grid-based docking search. For complexity comparisons with other grid based methods, please see [30].

RMSD Given two vectors $\mathbf{a}, \mathbf{b}$, each containing N points in 3D, the RMSD between them is defined as

$$\sqrt{\left(\frac{\sum_{i=0}^{N}((\mathbf{a}[i][0]-\mathbf{b}[i][0])^2+(\mathbf{a}[i][1]-\mathbf{b}[i][1])^2+(\mathbf{a}[i][2]-\mathbf{b}[i][2])^2)}{N}\right)}$$

5.1.2 Redocking

In redocking, the two proteins taken from the bound complex are computationally docked. From the list of 71 complexes, 1E96.pdb and 1F51.pdb could not be assigned charges using APBS [19] and were not considered any

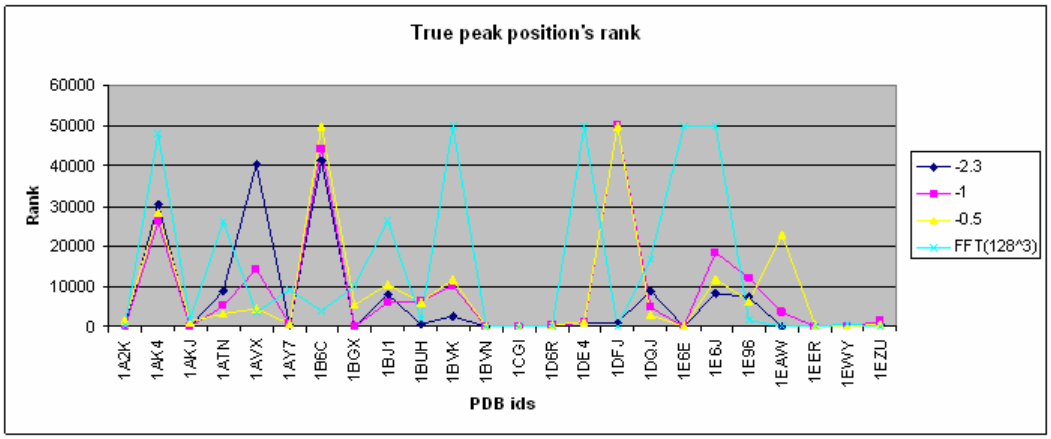


Figure 5.3: Comparison with docking with various rates of decay using 12 degrees rotational sampling, 32 fourier coefficients and a 128^3 FFT

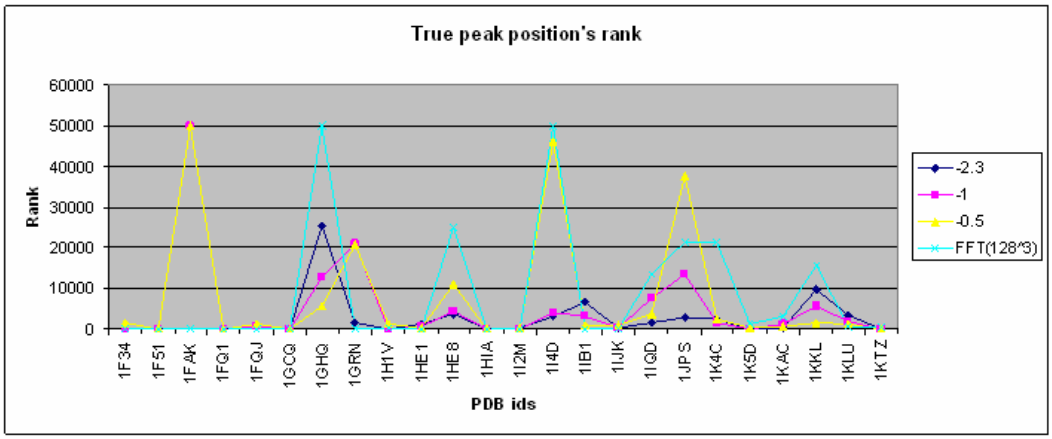


Figure 5.4: Comparison with docking with various rates of decay using 12 degrees rotational sampling, 32 fourier coefficients and a 128^3 FFT

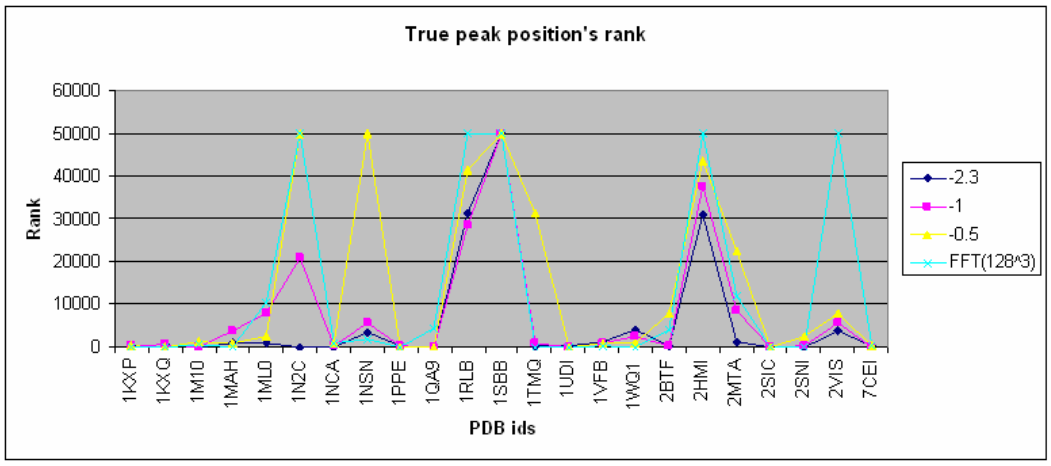


Figure 5.5: Comparison with docking with various rates of decay using 12 degrees rotational sampling, 32 fourier coefficients and a 128^3 FFT

further. Our results for redocking, using shape complementarity is shown in tables 5.4, 5.5. We used a rotational sampling of 20°. The number of Fourier coefficients were around 32^3, which is seen to retain around 95% of the energy in the docking profile. From these two tables, we see that we were able to predict good peaks in the top 2000 for 38 complexes, and could not predict any good positions for 1VFB.pdb, 1EER.pdb, 1E6J.pdb and 1HE8.pdb. We present the results sorted according to the surface area of the complex. It is interesting to note that smaller surface areas resulted in more number of good predictions. To compute the RMSD, we used all atoms of the ligand in the interface. The cutoff RMSD used was 5Å. But 22 complexes were found within 2ÅRMSD, 47 within 3Åand 58 within 4Å. In this thesis, we do not further refine the search from these set of peaks to predict an actual complex using energetics. We provide the time taken to compute the the docking profile in

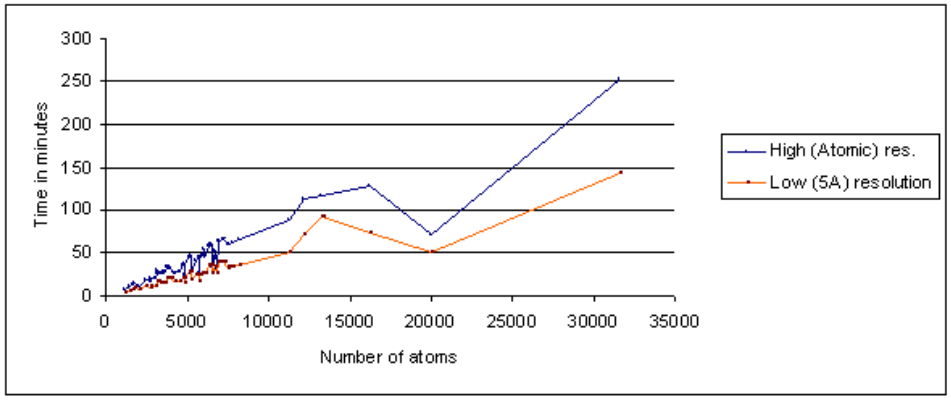


Figure 5.6: Time taken for docking using atomic and lower resolution models. The X axis represents the maximum of number of atoms in either protein.

figure 5.6. Rotational sampling for the remaining experiments were obtained (from a quaternion sampling algorithm) courtesy of Dr Olson's lab at TSRI, USA.

5.1.3 Bound-unbound Docking

Since we are interested in flexible protein-protein docking, we first consider the effectiveness of soft docking on bound-unbound docking. For this set of experiments, we take one protein from the docked complex and a known independent structure of the other protein. On doing an analysis of the interface of the files obtained from APBS, we found that 1EAW, 1BVK, 1ATN, 1EZU, 2BTF, 2VIS, 1KXP and 1BGX had too many missing atoms in the region. Hence, we removed them from the list of 69 complexes and present results for the remaining in tables 5.6 and 5.7. We used the same parameters as the redocking case. Out of the 60 complexes tested, soft docking found

PDB ID	Rank	N P	Best RMSD	PDB ID	Rank	N P	Best RMSD
1GCQ	12	22	2.159996	1AVX	6931	5	2.442265
1AY7	484	20	1.664378	1BUH	282	7	2.709604
1PPE	15	12	1.703695	1GRN	167	6	2.641065
1KTZ	539	22	2.008881	1EWY	793	11	2.065424
1QA9	1877	7	2.355009	1F34	1	3	3.562068
7CEI	2419	12	1.297948	1B6C	154	9	2.197851
1D6R	114	9	2.007833	1IJK	4423	4	1.838018
1HIA	249	12	1.545003	1BVN	827	8	2.572958
1CGI	53	9	1.96838	1A2K	2165	6	2.092011
1EAW	73	4	2.380758	1TMQ	1624	7	1.992915
2SNI	0	12	1.556862	1GHQ	1611	2	3.700142
1UDI	262	10	2.188662	1M10	1029	1	4.026118
1KAC	5967	8	1.685013	1FQJ	395	6	2.61232
2SIC	22	16	1.81577	1I2M	67	15	1.903976
1HE1	2	12	1.023882	1WQ1	22	3	3.372812
1VFB	-	0	5.527322	1KXQ	105	2	2.524465
1AK4	1146	11	1.148174	2BTF	21725	1	3.261152
1BVK	6957	1	4.543537	1MAH	7466	1	4.040359

Table 5.4: Protein-protein redocking results using shape complementarity ..1. 'Rank' is the best rank among all predicted positions whose RMSD was less than 5Å. 'N P' is the number of peaks in the predicted set which were less than 5ÅRMSD from the known position. 'Best RMSD' is the lowest RMSD among all the peaks that were shortlisted. If there were no good predictions in the top 50,000 that we choose to keep, we enter a '-' in the column for 'Rank'. The proteins are ordered by their surface areas.

PDB ID	Rank	N P	Best RMSD	PDB ID	Rank	N P	Best RMSD
1FQ1	933	7	2.005644	1EER	-	0	8.973606
1DFJ	0	11	1.80656	1RLB	13822	2	1.174494
1SBB	3139	10	1.528597	1IB1	3910	2	3.451666
1DQJ	4432	2	1.907411	1AKJ	886	7	1.110888
2MTA	2266	5	2.753511	2VIS	16781	4	2.465282
1EZU	39397	1	2.363247	1K5D	120	4	2.453245
1IQD	10752	5	3.519911	1H1V	1962	4	1.288332
1K4C	11343	4	2.095662	1E6J	-	0	5.554778
1FAK	17	6	1.588706	1NCA	5692	5	2.245905
1E6E	18	5	2.605772	1ML0	32582	1	4.319139
1ATN	2815	5	1.767566	1KXP	236	4	3.785924
1NSN	31399	1	4.938059	1HE8	-	0	6.315094
1KKL	11	2	1.919542	1BGX	43	1	3.330236
1I4D	17649	2	4.391554	1DE4	28305	2	3.448057
1JPS	1327	3	2.838963	2HMI	47242	1	3.468522
1KLU	11294	7	2.061819	1N2C	4929	2	4.784001
1BJ1	17946	1	3.221278				

Table 5.5: Protein-protein redocking results using shape complementarity..2. 'Rank' is the best rank among all predicted positions whose RMSD was less than 5Å. 'N P' is the number of peaks in the predicted set which were less than 5ÅRMSD from the known position. 'Best RMSD' is the lowest RMSD among all the peaks that were shortlisted. If there were no good predictions in the top 50,000 that we choose to keep, we enter a '-' in the column for 'Rank'. The proteins are ordered by their surface areas.

peaks (within 5ÅRMSD) for 23 within the top 2000 predictions, 39 within the top 10000, 53 within the top 50000 and failed for 8 cases (1GCQ, 1I2M, 1DQJ, 1FAK, 1EER, 1ML0, 2HMI and 1N2C). Unlike redocking, we get only 2 complexes with peaks within 2ÅRMSD of the actual, 23 within 3Åand 42 within 4Å.

PDB ID	Rank	N P	Best RMSD	PDB ID	Rank	N P	Best RMSD
1GCQ	-	0	-	1AK4	354	8	2.257936
1AY7	2758	14	2.579163	1AVX	42215	1	4.285829
1PPE	61	22	2.823658	1BUH	773	5	2.471546
1KTZ	12874	10	3.418981	1GRN	3218	1	4.973951
1QA9	3192	3	4.286313	1EWY	446	8	3.288623
7CEI	99	15	2.663766	1F34	4	2	3.923744
1D6R	4383	9	2.399734	1B6C	571	5	2.589875
1HIA	2902	28	2.943805	1IJK	9540	3	3.40416
1CGI	9488	1	4.497934	1BVN	1382	3	4.057453
2SNI	2	8	1.637314	1A2K	1137	3	3.538182
1UDI	15306	5	3.524703	1TMQ	499	2	2.616295
1KAC	6659	4	3.198683	1GHQ	16520	4	4.210506
2SIC	9	10	2.057806	1M10	11367	2	3.410315
1HE1	1	9	2.770525	1FQJ	5862	6	3.556944
1VFB	24581	2	3.448647	1I2M	-	0	-

Table 5.6: Bound-unbound docking results using shape complementarity..1. 'Rank' is the best rank among all predicted positions whose RMSD was less than 5Å. 'N P' is the number of peaks in the predicted set which were less than 5ÅRMSD from the known position. 'Best RMSD' is the lowest RMSD among all the peaks that were shortlisted. If there were no good predictions in the top 50,000 that we choose to keep, we enter a '-' in the column for 'Rank'. The proteins are ordered by their surface areas.

PDB ID	Rank	N P	Best RMSD	PDB ID	Rank	N P	Best RMSD
1WQ1	34	5	2.422902	1JPS	9655	5	2.807827
1KXQ	86	2	2.694885	1KLU	16336	10	2.598834
1MAH	26	5	2.486886	1BJ1	17610	1	3.300814
1FQ1	345	3	3.59622	1EER	-	0	-
1DFJ	38	6	3.405731	1RLB	16708	3	3.499713
1SBB	491	7	1.878689	1IB1	1047	1	4.426045
1DQJ	-	0	-	1AKJ	9009	4	3.14449
2MTA	2489	4	2.255889	1K5D	4714	2	3.937634
1IQD	40224	1	3.324618	1E6J	14233	3	2.838202
1K4C	18016	6	2.286136	1NCA	24641	1	4.647226
1FAK	-	0	-	1ML0	-	0	-
1E6E	26	3	3.238776	1HE8	29673	1	3.445586
1NSN	214	45	2.157744	1DE4	30009	1	4.606551
1KKL	1722	3	4.509084	2HMI	-	0	-
1I4D	3036	1	4.605807	1N2C	-	0	-

Table 5.7: Bound-unbound docking results using shape complementarity..2. 'Rank' is the best rank among all predicted positions whose RMSD was less than 5Å. 'N P' is the number of peaks in the predicted set which were less than 5ÅRMSD from the known position. 'Best RMSD' is the lowest RMSD among all the peaks that were shortlisted. If there were no good predictions in the top 50,000 that we choose to keep, we enter a '-' in the column for 'Rank'. The proteins are ordered by their surface areas.

5.1.4 Electrostatics Interactions

Electrostatics based affinity function was defined earlier in §4.2.3 using a model by Gabb [60]. We add this term to our docking score and tabulate the new results in tables 5.8 and 5.9. For each complex, we used a cutoff of 5Åas the RMSD required between the locations of ligand interface atoms in the predicted position vs the known crystal structure. This time, we use a more reasonable cutoff of 4000 positions only. Similar values of 4000, 4000-7000 have been cited in [27, 56]. We see that adding electrostatics enables us to get hits in the top 4000 positions, reducing the computations required in finer docking stages.

5.2 Flexible Docking

For all the previous set of complexes, we have computed a better side chain fitting at interfaces using our side chain optimization algorithm. We also provide preliminary analysis of the flexibility of molecules and perform conformational sampling for soft docking.

5.2.1 Flexibility Analysis and Conformational Sampling

Our flexible docking algorithm involves global conformational sampling of large scale domain motions, soft docking and refitting of side chains at interfaces. Domain analysis using the algorithm in section §4.5.4.1 was performed on three different complexes. In figures 5.7,5.8 and 5.9, we show the flexible regions and rigid domains identified using Normal Mode Analysis of Domain-

PDB ID	Rank	N P	Best RMSD	PDB ID	Rank	N P	Best RMSD
1GCQ	788	5	4.410048	1AK4	2641	9	2.105113
1AY7	473	16	2.660349	1AVX	588	2	3.300274
1PPE	1677	12	2.858134	1BUH	299	6	2.857323
1KTZ	822	11	3.367324	1GRN	-	0	-
1QA9	41	7	3.695077	1EWY	463	9	3.258474
7CEI	1532	11	2.118427	1F34	643	3	3.879021
1D6R	1413	6	2.522827	1B6C	804	7	2.342338
1HIA	73	11	2.120566	1IJK	419	6	2.530123
1CGI	2974	1	4.757587	1BVN	1894	3	3.729099
2SNI	392	11	1.892177	1A2K	330	5	3.103228
1UDI	3603	8	3.123136	1TMQ	59	7	2.068443
1KAC	2615	7	2.097499	1GHQ	1636	2	3.876798
2SIC	58	15	2.332064	1M10	399	1	4.582629
1HE1	4	11	2.314517	1FQJ	577	5	3.346168
1VFB	-	0	-	1I2M	-	0	-

Table 5.8: Bound-unbound docking results using electrostatics and shape complementarity..1. 'Rank' is the best rank among all predicted positions whose RMSD was less than 5Å. 'N P' is the number of peaks in the predicted set which were less than 5ÅRMSD from the known position. 'Best RMSD' is the lowest RMSD among all the peaks that were shortlisted. If there were no good predictions in the top 4,000 that we choose to keep, we enter a '-' in the column for 'Rank'. The proteins are ranked by their surface areas.

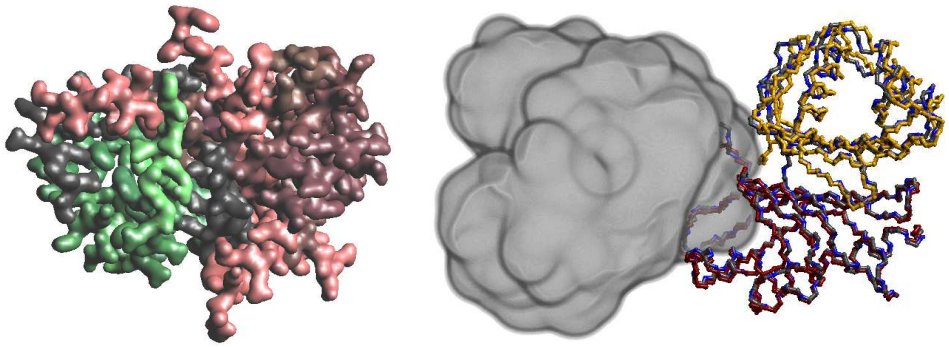
PDB ID	Rank	N P	Best RMSD	PDB ID	Rank	N P	Best RMSD
1WQ1	97	1	3.897003	1JPS	2538	3	3.095251
1KXQ	1492	2	3.168111	1KLU	124	6	2.75031
1MAH	1094	1	4.654828	1BJ1	1121	1	3.071451
1FQ1	52	3	3.168911	1EER	-	0	-
1DFJ	288	7	2.555477	1RLB	2182	2	2.293018
1SBB	1011	7	2.157762	1IB1	1	2	4.703256
1DQJ	1923	1	3.19577	1AKJ	2159	1	2.951469
2MTA	627	5	2.723326	1K5D	266	2	4.417297
1IQD	120	4	3.334374	1E6J	-	0	-
1K4C	783	4	2.105856	1NCA	139	10	2.801303
1FAK	-	0	-	1ML0	-	0	-
1E6E	135	4	3.228511	1HE8	-	0	-
1NSN	1016	1	4.203168	1DE4	-	0	-
1KKL	2421	1	3.711123	2HMI	-	0	-
1I4D	1010	2	4.441025	1N2C	-	0	-

Table 5.9: Bound-unbound docking results using electrostatics and shape complementarity..2. 'Rank' is the best rank among all predicted positions whose RMSD was less than 5Å. 'N P' is the number of peaks in the predicted set which were less than 5ÅRMSD from the known position. 'Best RMSD' is the lowest RMSD among all the peaks that were shortlisted. If there were no good predictions in the top 4,000 that we choose to keep, we enter a '-' in the column for 'Rank'. The proteins are ranked by their surface areas.

Finder and our clustering. We look at three primary motion types: shear, hinge and a combination of both. Shearing motion is shown at the large interface in 1A2K.pdb, the combination in 1VFB.pdb and a severe hinge motion in calmodulin 2BBM.pdb. Below we provide results from adaptive sampling to see how close we can get to a bound conformation from an unbound one using our model. We check each using shape based docking. Rotational sampling at hinge bending axes were computed using a Euler angle sampling.

Computing RMSD: To measure RMSD, we use the algorithm by [89] implemented in VMD [77]. They first do a best fit alignment of the molecules, given a common set of atoms in each and then do a RMSD calculation, again using any common set of atoms specified by the user.

GDPRan-NTF2 Complex (1A2K.pdb): In the docking set given, only the C chain from the C,D and E chains of RAN is given. The A and B chains of the NTF2 protein are used as the flexible protein and domain analysis followed by adaptive conformation sampling is performed on it. From DomainFinder, we compute two domains with 88 and 57 residues. Using the above model, we build our FCC with the following information: The interface area at the flexor measures 436.612 $Å^2$. The flexor is a cut of our FCC. Due to the large interface area between the domains of the ligand at its only flexor, it is given a shear motion. Although we have a cut, the large interface limits the angle of search (This also prohibits any twisting motion in our model.). In figure 5.7(a),



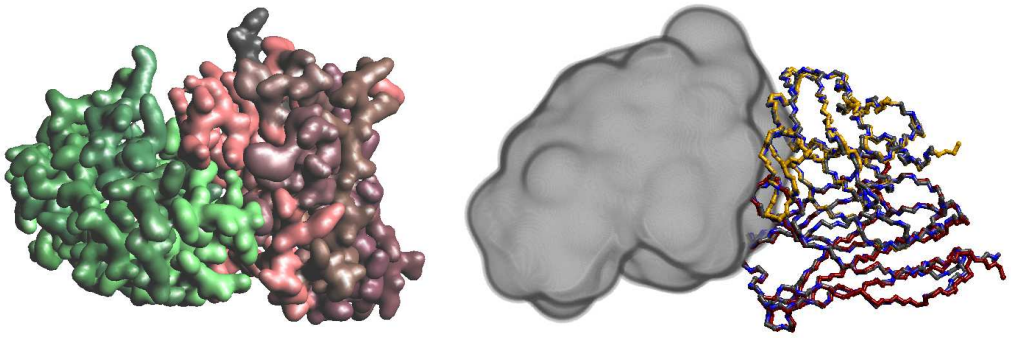
(a) Two large domains of NFT2 protein consisting of 88, 57 residues, 12,8 rigid segments and 10, 6 flexible loops are shown in green and red. The darker shades represent the rigid domains while the lighter shades the flexible loops. There are also 3 connectors of 7,8 and 12 residues shown in black.

(b) The GDPRan protein is shown in gray transparency at 5 Åresolution while the bound NTF2 ligand's backbone is in blue. The two chains of the unbound structure are shown in gold and red. The original RMSD between these bound and unbound structures is 1.453Å.

Figure 5.7: GDPRan-NTF2 Complex

we show the NFT2 protein colored by the domains. In the right hand side, in figure 5.7(b), we overlap the bound and unbound proteins. To compute RMSD and to fit the proteins, we used backbone atoms from residues 4 to 126 from chain A and residues 4 to 124 from chain B. Using our FCC model and adaptive conformational search, we get unbound structures with RMSD ranging from 1.111 Åto 2.06 Åto the bound structure. The unbound crystal structure has a RMSD of 1.453 Åto the bound protein. Hence we can get conformations closer to the bound structure using our FCC sampling.

Immunoglobulin-Hen egg white lysozyme Complex (1VFB.pdb): The A B chains of the immunoglobulin is used as the flexible protein docking to the



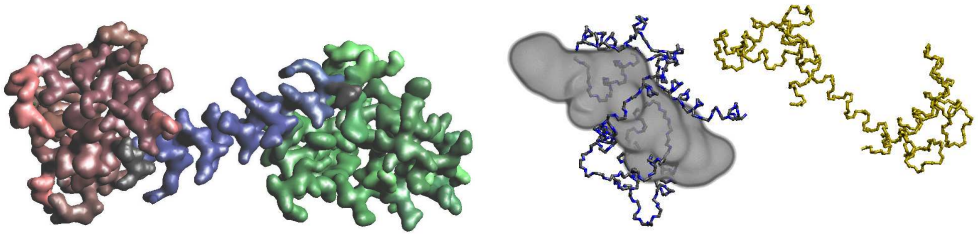
(a) This consists of 2 distinct domains shown in green and red with 67, 64 residues, 12, 9 rigid segments and 11, 9 flexible loops. The single connector has 6 residues in it.

(b) The colors represent the same structures as in figure 5.7. The original RMSD between the bound and unbound structures is 3.784 Å.

Figure 5.8: Antigen-lysozyme Antibody Complex

lysozyme. After domain analysis, we obtain two large domains at level 0. The flexor of this model is not associated with any shear as the interface area between the domains of the flexible immunoglobulin is only 101.433 $Å^2$. It is also a cut of the FCC and hence allows large bending motion at its hinge. Using this FCC model and rotations, we adaptively compute a set of conformations for the immunoglobulin. All backbone atoms of chain B and of residues from 1 to 107 of chain A were used in RMSD and fitting. We obtain RMSDs ranging from a high of 24.902 to a low of 0.586 Å. The original RMSD between the unbound and bound immunoglobulin is 3.784 Å. Hence we see a significantly closer conformation by our sampling.

Calmodulin bound to kinase (2BBM.pdb): The first two examples were picked at random from the docking set in tables 5.8 and 5.9. We also decided



(a) Calmodulin consists of two domains (in red and green) linked by a third in blue through connectors in black. They have 55, 51, 18 residues and 4, 5, 0 flexible loops respectively.

(b) This molecule has been well studied due to its large conformation change on binding. Especially note the breakage of the central helix. We separate the bound and unbound structures to show the large conformational change.

Figure 5.9: Calmodulin with a kinase

to present calmodulin as it is known for its large conformational change. In this complex (2BBM.pdb), we use calmodulin and a target peptide given by a myosin light chain kinase. We let calmodulin be the flexible protein and to test the accuracy of Normal Mode Analysis based domain finding, we again use DomainFinder to compute domains. We obtain 3 domains for calmodulin. One of them contains the central helix. But from figure 5.9, we see that the central helix breaks into two during its conformational change for binding. Hence, we were unable to get a close RMSD to the bound protein from the unbound calmodulin (from 1CLL.pdb). The RMSD was computed using backbone atoms of residues 4 to 147. The RMSD of the unbound state was 14.429 Å!. The best RMSD found from rotating about the two flexors between domains 1,2 and domains 2,3 was just 8.590. Hence we see that for large conformational changes, user input or analysis of more than one conformation is necessary.

5.2.2 Side Chain Optimization

We perform side chain optimization using algorithm 4.5.4.1 for each of the complexes listed in tables 5.8 and 5.9. We again use a cutoff of 5Åfor denoting a hit. We consider all positions, where our scoring function was at least 90% of the original score. This time, we see hits for 1N2C.pdb and 1ML0.pdb. The hit for 1ML0.pdb was surprisingly ranked first. We still could not dock 10 complexes (out of 63).

5.3 Summary

Our non-equispaced fast Fourier based docking algorithm gave successful results for 53 of 63 complexes. We use on the average, less than 30 mins per complex on a single CPU (1.6GHz), with 512MB RAM laptop. We also have a parallel implementation on different architecture machines. Electrostatics improved the scoring dramatically as compared to just using shape complementarity. Flexibility models of proteins were created using Normal Mode Analysis. This was used to compute a diverse set of conformations which can be used in docking and effectively sampling orientation space. A finer refitting of side chains at interfaces had the effect of getting two more successes from the list of complexes. 1N2C.pdb and 1MLO.pdb, which could not be docked using soft docking were docked by just improving the position of large side chains. All of these interactions can be better studied by visual inspection of surfaces, functional properties and interfaces. In the next chapter, we provide new algorithms for high quality visualization of large proteins in a multiresolution

manner.

Chapter 6

Visualization

Scientific visualization of proteins, their properties and interactions is crucial in understanding the complex set of phenomenon that characterize any life process at the molecular and cell level. Both scientific discovery and data analysis is enabled through real time, high fidelity visualization of such interactions. Visualization, animation and visual analysis of such interactions have been studied along with the synthesis of crystal structures, new molecular surface definitions and analysis of protein interaction networks. Realistic molecular systems include more than millions of interacting atoms in a dynamic environment. Hence, high performance visualization algorithms are a necessary tool to study protein structure and interactions and forms the last part of this thesis.

6.1 Molecule Visualization and Protein Docking

Images and animations of protein interactions, both precomputed and real time simulations have been used to provide qualitative and quantitative feedback to users. In [2], different molecular surfaces and the alpha complex defined by Edelsbrunner are visualized in a CAVE environment. VIBE [40], a

VIrtual Biomolecular Environment by Cruz-Neira, Langley and Bash is a system based on the CAVE environment. A user is immersed in a virtual world, and allowed to interact with a protein and ligand, while molecular dynamics is running on a parallel back end. In their paper, they used the example of HIV protease-cyclic urea inhibitor complex and the CHARRMM molecular dynamics program [25]. The user interaction allows a larger sampling of space than possible with just molecular dynamics. Levine et. al. in [109] introduced the program STALK, where a user, again in a CAVE environment is allowed to initialize the search position for a genetic algorithm to perform rigid protein-protein docking. To improve the rather cumbersome interactivity in current VR systems for docking, Anderson and Weng introduced VRDD (Virtual Reality visualization to protein Docking and Design) in [5]. They tested their environment on three complexes: nine residue peptide - MHC, barstar - barnase and antibody-hemagglutinin with partial success. VMD [77] has been tightly coupled with NAMD, a molecular dynamics package to allow users to both visualize and steer the dynamics.

6.1.1 Requirements

Since protein interactions involve a highly complex and dynamic scene with millions of atoms, the visualization requirements are quite challenging:

1. **Interactivity:** Molecular environments and protein interactions are dynamic in nature and require interactive visualization, where users can visualize, steer and receive quantitative and qualitative feedback from

the system.

2. **High visual quality:** Current systems do not provide high quality interactive visualization due to the inherent complexity in rendering scenes with curved surfaces. Traditionally, graphics cards and associated software have been optimized for rendering scenes with triangulated geometry: a bad choice for the curved atoms and molecular surfaces.

3. **Scalability:** As we move from simple docking of small proteins to large molecular assemblies and cells which contain billions of atoms, scalable visualization becomes a necessity.

4. **Representation of error and multiple functions:** Molecular imaging data and simulations are associated with noise and errors. Depiction of these errors in visual rendering has not been done and often leads to misleading images. Apart from shape, electrostatics, hydrophobicity, h-bonds, atomic contact energies, atom types, residue types etc are multiple functions that users are interested in. It is a challenge in computer graphics to visualize, harmoniously, such a large number of functions in one scene.

With these goals in mind, we have developed several visualization algorithms, and summarize two of them below.

6.2 Imposter Based Schematic Model Rendering

While molecular visualization software has developed over the years, today, most tools still operate on individual molecular (protein or RNA - *Ribonucleic acid*) structures and small electron charge and electrostatic potential fields, with little facility to manipulate larger multi-component complexes, integrate geometric and volumetric visual representations, or effectively depict molecular flexibility and dynamics. Few, if any currently used programs allow for or enable interaction with multi-component macromolecules and their atomic level properties, such as reconstructed volumetric maps from tomographic and cryo imaging, that will become common in the next five to ten years. We introduce a new imposter rendering based algorithm which allows orders of magnitude speedup with an increase in visual accuracy, allowing users to interactively visualize large interacting molecular assemblies.

6.2.1 Related Work

Molecular Modeling and Structural Rendering Numerous modeling schemes have been used to represent and display molecules and their properties on computers [103]. Some models which are structural in nature include the Stick model, the Ball-and-Stick model, the Wire-Frame model and the Cartoon model. All these in fact are different visual representation of an underlying hierarchical skeletal model of the positions of atoms, bonds, chains, and residues in the molecule. Hence, structural models are designed to represent the primary, secondary, tertiary and quaternary geometric structures of

the molecule. Many visualization systems such as RasMol [148], Chime, Protein Explorer [117], PyMol [43], VMD [77], MidasPlus [57], PMV have been created to display the three-dimensional structure of a molecule in different styles. A complementary approach to modeling molecule structure is to model properties of the molecule as 3D fields defined over the structural model. This enables either extracting isosurfaces of the fields and their derivatives or examining them using other visualization techniques such as volume rendering and topology graphs.

Molecular Surface Rendering One of the early visualization programs [105] by Lee and Richards, in 1971, used arcs of circles (intersections of the atoms with a series of parallel cutting planes) to represent the contour of both van der Waals surface and the Solvent Accessible Surface. After the introduction of the Solvent Excluded surface by Richards, Connolly, in a series of papers (see [36][37]) introduced both analytical expressions for the patch complex and dot based representations. In [36], he provides visualization of the SES surface of insulin. A summary of early work on rendering van der Waals surfaces and SES surfaces is presented by Max in [118], who also added more visual features like shading to them. The patches were represented in NURBS format to allow for both modeling and rendering in [14] and extended to dynamic surface construction with varying probe radii in [16]. Blinn [22] used Gaussians to render atoms. Apart from structure, functions like gradients of density and electrostatics were rendered in [138]. Much of this preliminary

work focused on finding fast methods of triangulating the solvent-accessible or excluded surface. Two prominent obstacles in surface visualization are the correct handling of surface self-intersections to avoid visual artifacts and the high communication bandwidth needed when sending tessellated surfaces to the graphics hardware.

Image Based Rendering Techniques Large environments have traditionally been rendered using image based rendering techniques. These techniques usually suffer from popping artifacts and are rigid to changes in lighting. Hence, a renderer must strike the right balance between visual fidelity and performance. View dependent texture mapping is commonly used for image-based rendering. In [42], the authors describe an efficient way of implementing such texture mapping through the traditional graphics pipeline. A survey of image based techniques for improving the rendering quality of traditional techniques is given in [31]. Large scale environment visualization through images was shown in [3, 69, 106, 149]. Better depth maintenance was shown in [41, 83]. In [46], an image based rendering algorithm is given which depends on lighting conditions. Using artistic techniques and multi-perspective images, [73] present a method of improving the occlusions of objects during motion.

Programmable Graphics Hardware The recent interest in image-based rendering has been due in large part to increasingly powerful and increasingly programmable graphics hardware. Cg is a high level Graphics Processing

Unit (GPU) programming language developed by NVIDIA that allows users to write custom programs for the vertex and fragment processors in commodity GPU's. Previous work for practical applications of GPU's [55] describes the texture-based sphere like patch rendering.

Our fast, general visualization technique overcomes the bandwidth and artifact problems associated with triangle-based structural rendering. Under imposter-based rendering, each complex geometric primitive is represented by a set of simpler geometric objects, such as a single quadrilateral, in contrast to the numerous triangles needed to tessellate complex curved primitives. In addition, primitive-primitive intersections under imposter-based rendering are correct on a pixel level with no extra computation, which differs from the expensive clipping algorithms needed to remove artifacts from intersecting sets of triangulations.

6.2.2 Internal Representation

The Flexible Chain Complex (FCC), introduced in section 3.1.3 is our main hierarchical representation for molecular structure. For each chain in the molecule, the chain backbone is stored along with all the residues (amino acids or nucleotides), which are placed at the connection point to the backbone. FCC allows the application to interactively update flexible dihedral angles along the backbone.

Physical properties of a molecule (e.g. the electron density, electrostatic potential, hydrophobicity, and surface curvatures) can be represented

as functions of the atom centers along with the atoms' properties. Hence the *FCC* provides a unique skeletal and an implicit volumetric format to efficiently represent structural and functional properties of a molecule. The *FCC* can also contain pointers to volume files which explicitly represent properties of the molecule. The volume files are in two formats. Apart from the conventional color index format, we also use the older *RGBA* format to efficiently render pre-computed structures of the molecule. For example, we could have an *RGBA* dataset where the molecule is colored per atom according to the color of the residue to which it belongs. The *A* channel is used to show the dropoff in electron density as distance from the atom center increases.

6.2.2.1 Static Level-of-detail

The *FCC* sorts the atoms and superstructures of a molecule into a four-tiered hierarchy which is derived from the biochemical hierarchy used in Protein Data Bank (PDB) files. The following is a top-down view of the hierarchy.

- Chain - chain backbone information is stored
- Secondary structure - a Helix, a Turn, or a Sheet
- Residue - either an amino acid or a nucleotide
- Atom - lowest tier, each member is a single atom

The hierarchy is constructed from the bottom up, so that every member of the hierarchy, except chains, contains all relevant members from the level immediately below it. For example, each atom only contains itself, since atom is the lowest level of the hierarchy. A residue contains all atoms that are a part of it, while a secondary structure contains all residues that are a part of it. Chains are a unique in that they only contain some of their component members, which will be explained in detail later in this section.

Each level of the hierarchy is associated with a geometric representation. An atom is represented by a single sphere with the atom's van der Waal radius. A residue is represented by a minimal single bounding sphere that encloses all of its component atoms. In Figure 6.5, we show the bacteriophage virus' capsid proteins (1GW7.pdb) at two different levels using different visualization techniques.

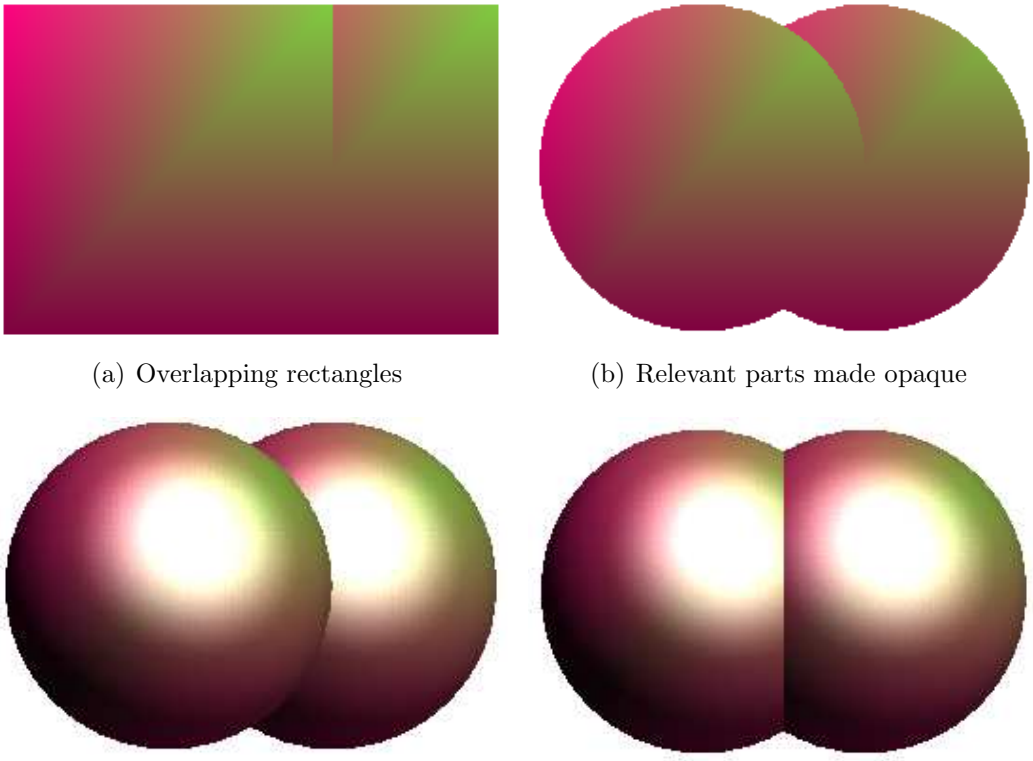
For secondary structures, sheets and helices are represented by an appropriately-sized geometric sets of cylinders and helices. The orientation, length and radius of these are determined by a least mean square fitting of the central axis to all of the atom centers in either the helix or the sheet's strand. The third type of secondary structure, turns, are simply represented by their component atoms and bonds along the chain backbone, since turns are usually short atom sequences that feature sharp angles. Not all residues are part of secondary structures, so we simply cluster these otherwise parent-less residues under an imaginary SS_NULL group.

Chains are represented by including only the atoms in the backbone of the chain. Hence, the chain level does not include any of the functional groups attached to the molecular backbone. This gives us a static Level Of Detail (LOD) for visualizing molecules are different length scales.

6.2.3 Imposter Rendering

The models we encounter in molecular visualization, including the union of balls model, the ball and stick model, the secondary structure model, are composed of regular curved surfaces. The traditional method of rendering these surfaces is to triangulate the surface and render the triangle strips. Due to the large sizes of the data sets we are trying to render, triangulated curved surface rendering is either too computationally expensive or is plagued by visual artifacts.

Hence we introduce some new image-based rendering techniques that extend earlier work in sphere rendering. Our main idea comes from the fact that in the programmable graphics hardware, we are allowed to perform a depth replacement at the fragment level. This allows us to render a curved surface with some axis of symmetry without the need for triangulation. NVIDIA's *Cg* Tutorial [55] describes a method of rendering spheres with a single quadrilateral, with correct depth and normals under an orthographic projection. In the perspective view, the depth turns out to be an approximation, but still retains good smoothness properties. We extend this work to include other primitives (Figure 6.2, 6.1) and apply these new primitives in our molecule



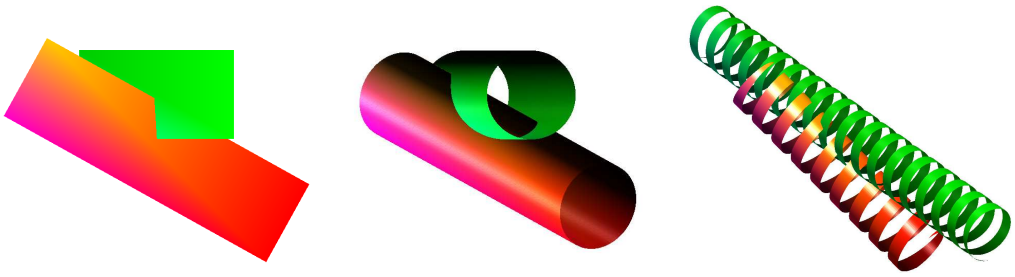
(a) Overlapping rectangles

(b) Relevant parts made opaque

(c) Spheres with normals and shading

(d) Spheres with depth and normals

Figure 6.1: Sphere rendering from NVIDIA's depth replace example. We consider two overlapping specular spheres, whose *correct* appearance is as shown in the rightmost image. We render two rectangles (leftmost image), perform opacity culling (left middle), perform per-pixel normal and lighting evaluation (right middle), and finally replace depth values to obtain the final image (rightmost image).



(a) Four rectangles (two overlapping pairs)

(b) Cylinders after correct opacity, lighting, and depth are assigned

(c) Helices formed from the previous pair of cylinders (new viewpoint).

Figure 6.2: Imposter rendering for bonds and helical structures.

visualization.

One key advantage of imposter-rendering is per-pixel accurate shading within the resolution of the normal map. Although the resolution of the normal map imposes some limits, the normal map's resolution may be increased arbitrarily based on the application requirements. In contrast to imposter-based renderers, typical molecular visualizers, such as RasMol and its derivatives, tessellate their geometric primitives, which produces approximate lighting effects and tessellation-related artifacts. Another benefit of imposter-rendering is its occurrence near the end of the rendering pipeline, meaning it can be combined with software-based acceleration techniques, such as occlusion culling and adaptive tessellation, to further increase visualizer performance.

6.2.3.1 CPK Model

The CPK model of a molecule is the geometric union of all of van der Waal radius spheres that correspond to the atoms in the molecule.

Sphere rendering Texture-based sphere rendering is shown in one of NVIDIA's *Cg* Tutorials [55]. We extend their method to render spheres. The cylinder and helix renderers that will be described in Sections 6.2.3.2 and 6.2.3.3, respectively, are extensions of the sphere renderer.

6.2.3.2 Ball and Stick Model

The ball and stick model is the geometric union of all the atoms and bonds in a molecule, where each atom is represented by a van der Waal radius sphere and each bond is represented by a cylinder that extends between the two atoms that form the bond. Atom radii must be intentionally reduced from the values given in the PDB file in order to make the bonds visible. While the ball and stick model is otherwise very similar to the CPK model, the understated atomic radii places more emphasis on the internal connectivity of the molecule.

Cylinder rendering The cylinder uses a single quadrilateral as the underlying geometric primitive, along with three texture maps. Unlike a sphere, a cylinder is not rotationally invariant, so its corresponding quadrilateral in clip space reflects the orientation of the actual cylinder in clip space. In our algorithm, we need only one quad to obtain a perfect cylinder, but due to a compiler bug in *Cg*, we are forced to use two quads. The algorithm for the vertex program and the fragment program is given in Algorithm 3 and Algorithm 4. Our vertex and fragment programs work in GeForce FX cards and

beyond.

In each of the algorithm descriptions, we have tried to be as brief as possible and also use common abbreviations due to space constraints, without losing any information. ES, OS refer to *EyeSpace* and *ObjectSpace* respectively. MV, MVI, MVP, $Proj$, $MVPI$ are the OpenGL $ModelView$, $ModelViewInverse$, $ModelViewProjection$, $Projection$ and the $ModelView$ $ProjectionInverse$ matrices respectively.

By using the vertex and fragment programs in Algorithm 3 and Algorithm 4, we create a single *front sided* cylinder where the two cylinder bases are not well defined. This design increases performance with no loss in visual quality, since the cylinder bases are always hidden inside spheres in the ball and stick model.

In Algorithm 5, we create cylinders with defined bases by ensuring that we have the correct opacity at the ends of the cylinder. This algorithm requires the coordinates of one end point, the axis of the cylinder, and the $MVPI$ matrix. Due to space constraints, we present only the subset of the program that needs to be inserted into the Stick Fragment Program given in Algorithm 4. Figure 6.2 depicts how quads are transformed in the programmable graphics card into cylinders with proper depth and lighting. Algorithm 3 is still used as the vertex program.

Algorithm 3 Stick vertex program

1: Inputs are: [P1, P2, MV, radius, offset, MVI, MVP, Proj]
2: $[OS]axis \leftarrow P2 - P1$
3: $[ES]axis \leftarrow normalize(MV * [OS]axis)$
4: $h \leftarrow \frac{1}{[ES,xy]gradient} = \frac{1}{sqrt(1-[ES]axis.z^2)}$
5: $[ES,xy]unitAxis \leftarrow normalize([ES]axis.y, -[ES]axis.x, 0, 0)$
6: $[ES,xy]axis = [ES,xy]unitAxis * offset * radius$
7: **if** P associated with $P1$ **then**
8: **if** $ES.axis.z < 0$ **then**
9: $[ES]offset \leftarrow [ES,xy]offset + [ES]axis * h * radius * [ES]axis.z$
10: **else**
11: $[ES]offset \leftarrow [ES,xy]offset + [ES]axis * h * radius * [ES]axis.z * -1$
12: **end if**
13: **else**
14: **if** $ES.axis.z < 0$ **then**
15: $[ES]offset \leftarrow [ES,xy]offset + [ES]axis * h * radius * [ES]axis.z * -1$
16: **else**
17: $[ES]offset \leftarrow [ES,xy]offset + [ES]axis * h * radius * [ES]axis.z$
18: **end if**
19: **end if**
20: $[ES]offset \leftarrow MVI * [ES]offset$
21: **if** P associated with $P1$ **then**
22: $outP \leftarrow P1$
23: **else**
24: $outP \leftarrow P2$
25: **end if**
26: $HPOS \leftarrow MVP * (outP + offset)$
27: $radius \leftarrow radius * h$
28: $[z,w]center \leftarrow [MVP[2], MVP[3]].outP$
29: $[z,w]offset \leftarrow radius * [Proj[2], proj[3]].z$
30: Output: [[z,w](center,offset), HPOS, LightVector, Normal, EndPoints, normalized xy position]

Algorithm 4 Stick fragment program

Inputs are: [normalized xy position (xy), LightVector, Color, Ambient light]
$Normal \leftarrow Lookup(normalMap, (xy))$
$Depth \leftarrow Lookup(depthMap, (xy))$
$OutColor = CalculateOutputColor()$
$[z, w] \leftarrow Depth * [z, w]offset + 1 * [z, w]center$
$OutDepth \leftarrow z/w * 0.5 + 0.5$
Output: [color,opacity,depth]

Algorithm 5 Cylinder fragment program

Inputs are: [P1, axisDCs, length, MVPI, IN.xy]
$ndcPos \leftarrow (IN.x, IN.y, z/w, 1)$
$[OS]P = MVPI * ndcPos$
$y \leftarrow axisDCs.(([OS].P - P1).xyz)$
if $y < 0$ **then**
  $opacity \leftarrow 0$
else if $y > length$ **then**
  $opacity \leftarrow 0$
else
  $opacity \leftarrow 1$
end if

Algorithm 6 Helix fragment program

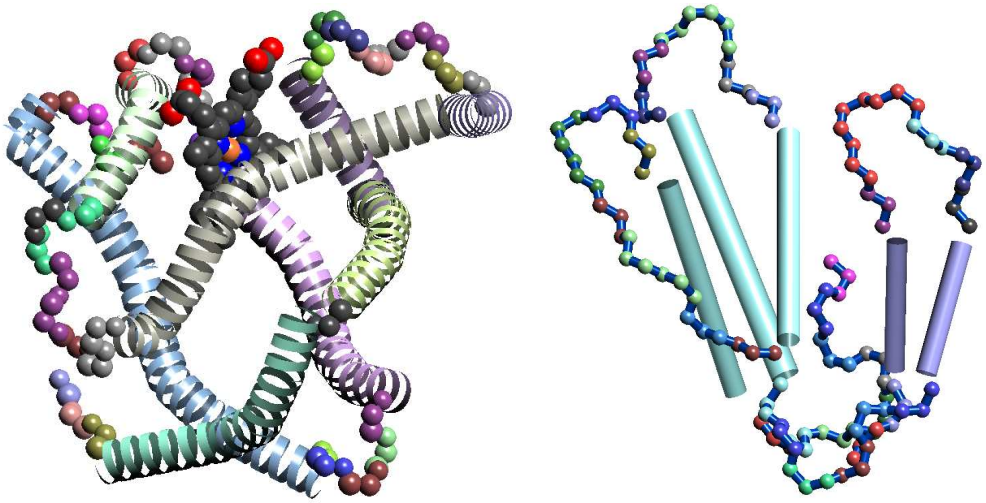
Inputs are: [refVector, crossRefVector, opacityMap, pitch]
if $y.inRange()$ **then**
  $V \leftarrow normalize((IN.xy) - P1 - y * axisDCs)$
  $x \leftarrow cos^{-1}(V.(refVector))$
  $side \leftarrow V.crossRefVector$
  if $side < 0$ **then**
    $x \leftarrow 2 * \pi - x$
  end if
  $y \leftarrow \frac{fmod(y,pitch)}{pitch}$
  $opacity \leftarrow lookup(opacityMap, x, y)$
end if

6.2.3.3 Secondary Structure Representation

We visually represent helix and sheet secondary structures using imposter-based helices and cylinders. Turns are represented by their component atoms and bonds along the backbone since they usually consist of only a handful of atoms.

Helix rendering Helices are composed of two quadrilateral primitives and are an extension to cylinder rendering. We present the fragment program in Algorithm 6, which is the extension which needs to be added to the cylinder fragment program in Algorithm 5. For this fragment program, we need the additional input parameter of a $crossRefVector$, which is a vector from one end point of the helix on the axis to any particular point on the helix in object space. Another input needed is an opacity map (a bitmap containing a wrapped diagonal band of the required thickness), which generates the shape of the helix via alpha values. Figure 6.2(c), shows the two cylinders from Figure 6.2(b) under the new helix opacity mapping. A slightly different viewpoint is used for the helices to show the correct per-pixel intersections and shading. Figure 6.3(a) shows the myoglobin molecule with its helices represented using imposters.

Sheet rendering Sheets are composed of multiple strands. One method of rendering a sheet is to render the set of strands as cylinders. Hence we use our cylinder rendering when visualizing sheets. The two β sheets of a snake



(a) Coils showing the presence of α helices in the myoglobin protein. The heme group is shown here with an orange iron atom.

(b) Sets of cylinders representing the strands of two β sheet in a snake toxin protein (1FSC.pdb).

Figure 6.3: Helices and sheets are rendered using imposter primitives.

toxin are represented by sets of differently colored cylinders in Figure 6.3(b). Multiple cylinders per strand and a color gradient could be used to enhance the folding of the sheet and the strand directions.

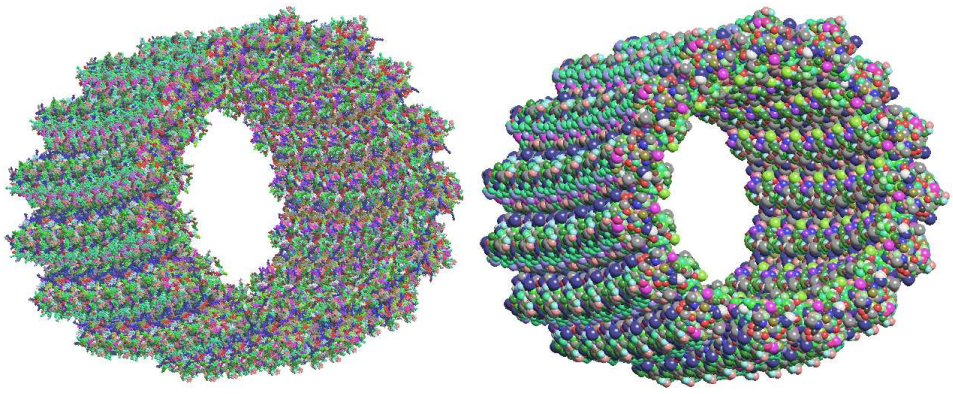
6.2.3.4 Function-on-surface Rendering

We implement embedded surface rendering by extending the previous GPU programs. It is often useful to study the variation of functions on molecular surfaces. These properties can be visualized by accessing two additional custom texture maps in the fragment program. The user can set one texture to the volumetric function, while assigning to the other texture a transfer function map which allows the user to interactively modulate the function being

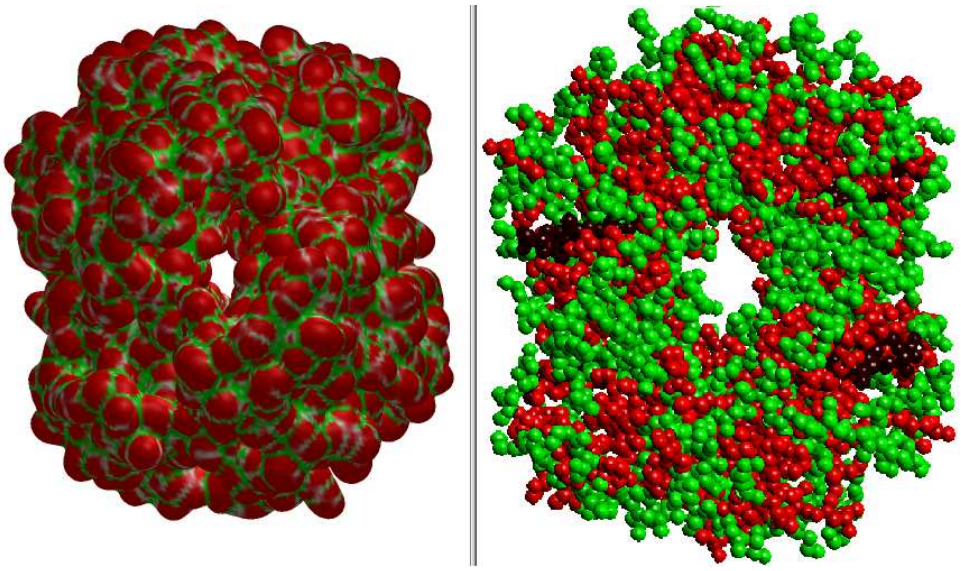
mapped onto the imposter model. One could also implement implicit functions using the fragment program instead of an explicit volumetric grid. In Figure 6.4(b), we show the hydrophobicity function being mapped onto the union of balls model of the hemoglobin molecule.

6.2.3.5 Dynamic Level-of-detail

Dynamic LOD uses the static LOD hierarchy to change regions of the molecule based on the spatial occupancy of these regions from the current viewpoint. Dynamic LOD is currently implemented with bounding spheres on all levels of the hierarchy to limit visually distracting popping when one region switches its LOD. When dynamic LOD is active, molecular regions that occupy fewer pixels on screen are represented with a coarser level in the static hierarchy. This allows the user to see atom-level detail by zooming in, while also being able to capture the overall molecular structure by zooming out. Additionally, realtime scenes with many molecules interacting with one another benefit from dynamic LOD by automatically reducing the geometric complexity of a scene from distant viewpoints, while maintaining the power to explore the scene in detail from other viewpoints.



(a) Imposter rendering of the 1.2 million microtubule molecule in two different LODs. The left image shows the atoms rendered as spheres with residue colors. The image in the right shows the residue level of the macromolecule. Rendering such large data sets in high resolution is especially useful for multi tile display and exploration of complex large macromolecules like the microtubule and large viruses.

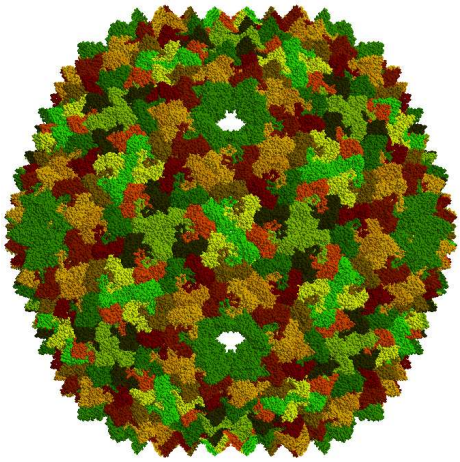


(b) A synchronized multi-view visualization of two functions of the same hemoglobin dataset. In the left pane, we show the mean curvature on an isosurface of the electron density of the molecule. The red color represents positive mean curvature, while the green shows negative mean curvature and white represents zero mean curvature regions. On the right pane, we have the hydrophobicity function mapped to the imposters, with red representing hydrophilic regions and green representing hydrophobic regions. The hemes groups are shown in black.

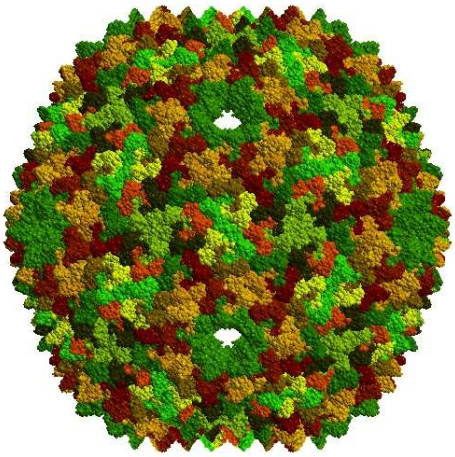
Figure 6.4: The efficiency of the rendering algorithm allows us to render large molecules at different resolutions and to display multiple synchronized views.

Mol	NumAtoms	NumRes	Mem (M)	LOD	FPS
Hem	4770	574	27	Atom	28.4
				Residue	21.3
				Chain	85.2
Rib	98543	6577	79	Atom	7.7
				Residue	9.5
				Chain	42.6
Mic	1210410	78210	643	Atom	1.9
				Residue	3.6
				Chain	2.4

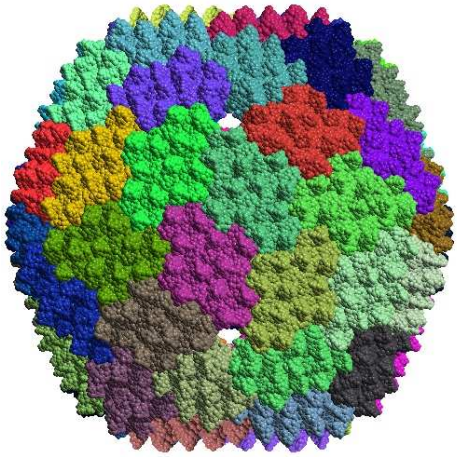
Table 6.1: Performance results for rendering in 800x600 resolution, full screen mode for Hemoglobin (Hem), Ribosome (Rib) and the Microtubule (Mic).



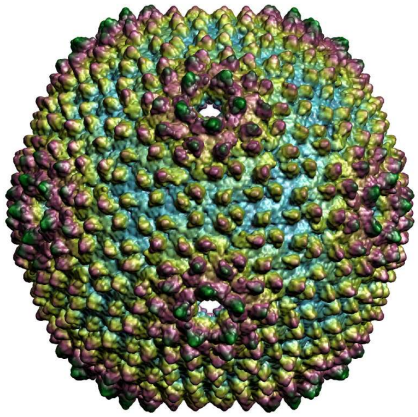
(a) Imposter rendering of atoms with chain colors



(b) Imposter rendering of residues with chain colors



(c) Imposter rendering of residues with protein colors



(d) Volume rendering at the residue level with depth coloring

Figure 6.5: The Bacteriophage PRD1 capsid protein (1GW7.pdb). The virus has 34181x60 atoms and 4452x60 residues. Figures (a) (b) and (c) show an imposter view of the virus at the atom level with chain colors and residue level with chain and protein colors. (d) is a depth colored image of the volume representing electron density

6.2.4 Results

Our implementations performance was measured on a dual Pentium III Xeon 800MHz machine with one gigabyte of memory and an NVIDIA GeForce4 Ti 4200 AGP8X for its GPU. Only one of the dual CPU processors was used during testing. Table 6.1 shows performance for the hemoglobin, ribosome, and microtubule datasets. The hemoglobin (1A00.pdb) and the large ribosomal subunit (1JJ2.pdb) are taken from the Protein Data Bank [172]. The microtubule data set was donated by Dave Sept and Nathan Baker. The second and third columns list the number of atoms and residues, respectively. The fourth column lists the peak CPU-side memory usage for each experiment. Since a given dataset is always processed the same way to generate the static LOD hierarchy, from which all static LODs can be derived, the CPU-side memory usage is the same regardless of which LOD is selected for rendering. The time taken to parse the files to a hierarchy is considered as a preprocessing step and is not considered here. When we are using dynamic LOD rendering, the time taken to determine the cut in the tree to render is negligible compared to the rendering times.

For each trial, the given molecule was rendered across three LODs (fifth column): atom level, residue level, and chain level. The final column lists the frames-per-second performance of the trial. All trials were performed at 800x600 resolution using a full-screen mode.

The atom LOD sends the most number of geometric primitives to the GPU, so cases where the atom LOD performs worst are vertex limited. We

expect this case to occur for extremely dense molecules with high atom counts. The residue LOD provides fewer geometric primitives to the GPU, but residue representations (a conservative bounding ball) take up more screen space than their component atoms. Thus, more fragments are generated in the GPU fragment processor, so cases where the residue LOD perform worst are fragment limited. Finally, the chain LOD reduces both vertex and fragment load from the atom LOD by 1) rendering only atoms and bonds along the backbone and 2) using primitives that are the size of atoms rather than larger bounding balls. Note, however, that the chain LOD renders more than one geometric primitive per residue, so it still has greater vertex complexity than the residue LOD, which renders exactly one geometric primitive per residue.

For the atom LOD, the hemoglobin visualization performs fastest. Since hemoglobin is a relatively small molecule, it stands to reason that it is not vertex limited. On the other hand, the medium-sized ribosome shows signs of being vertex limited, since its performance improves slightly under the residue LOD. Finally, the microtubule, which contains the most vertex information, shows a speedup of about 90% when switching from atom LOD to residue LOD. The slowdown of the microtubule chain LOD is a result of the vertex-limited nature of the rendering, since three atoms are still being drawn per residue. Memory usage for the two larger molecules scales linearly with the size of the molecule, while the hemoglobin memory usage is almost entirely due to the operating overhead of the program (about 20 M). The bacteriophage shown in Figure 6.5 renders at a peak rate of around 1.2 fps at the full atomic

level resolution.

6.2.5 Summary

Fast imposter-based rendering algorithms are introduced to render structural molecular shape at various LODs. This technique allows us to animate large molecules at high resolution that were previously intractable. The above algorithms have been implemented in a software called TexMol, described in [13]. Currently, the massive microtubule complex (Figure 6.4(a)) animates at a peak rate of four frames per second.

6.3 Interactive Rover

While the imposter rendering algorithms in section §6.2 allows users to interactively visualize very large molecules, including viruses, there is also the need for providing fast rendering algorithms for large molecular surfaces. Ribosomes, microtubules, viruses etc contain a hundred thousand to nearly tens of million of atoms. Interactive visual exploration of such large complex structures provides a wealth of information for molecular biologists. It becomes important to present both local features like active sites and global features like the interaction of tertiary motifs and the overall quaternary structure. We present level-of-detail molecular surface generation and rendering algorithms for interactive exploratory visualization of large biomolecular complexes. Our dynamic surface generation and exploratory visualization technique allows the user to explore these biomolecular complexes, at interactive speeds, with re-

gions of interest captured via a roaming microscopic cube, the size and position of which is controlled by the user. The molecular surface of these structures is defined as a level set of a function (see section §3.3) that is dynamically constructed by the summation of decaying atomic electron density kernels. Adaptive molecular surfaces are constructed by allowing the users to control both the grid resolution and kernel decay rates in the regions of interest. To achieve interactivity, tree based data structures are utilized for atomic center identification within a roaming microscopic cube and a fast Fourier based summation of kernel functions for local and global updates, coupled to adaptive crack-free contouring and rendering. Our dynamic multiresolution surface generation algorithm is shown to be scalable to very large complex biomolecules.

As the resolution of the function is varied, different features of the molecules are discernable. As the smoothness of the kernel is increased, we obtain lower resolution maps. At high resolutions, near the 1 and 2 Å range, we can distinguish atoms from each other, showing a very high detailed map of the molecule. At atomistic resolution, biologists can study the active sites in detail. Active sites, or sites where different proteins and ions interact, protein-ligand interfaces, bond structures are all studied by molecular biologists at this resolution. For example, the oxy-deoxy process in the hemoglobin molecule occurs around the heme region represented by just a few important atoms. At lower resolutions, secondary structures become more apparent in volumetric maps. Imaging data of large structures like icosahedral viruses, captured at very low resolutions show the arrangement of capsomers on their shell. When

users look at each capsomer in a higher resolution, the arrangement of proteins, then the structures in the proteins and finally the atomic arrangements become apparent. Hence we see that studying large biomolecular complex structures involves visualization at multiple levels of resolution. In figure 6.6, we show an example of visualizing the hemoglobin molecule with the active site at a higher resolution[1].

Sampling of high resolution images needs to be dense enough to capture atomic details. For molecular complexes involving millions of atoms like the microtubule and many viruses like the Rice Dwarf Virus (1UF2.pdb), Human Rhinovirus (1FPN.pdb), the sampling requires new memory management algorithms to handle the large volume maps. Switching from high to low resolutions in different regions and interactive exploration of large maps is not scalable. Isocontouring of these maps will yield more than hundreds of millions of triangles. Since the user is always interested in high resolution at local regions and global features at a coarser resolution, we present the idea of a ***Rover***. The *rover* is an exploratory microscopic cube, whose size and position is controlled by the user. Regions inside the *rover* are computed at high resolution in both kernel space and sampling space. The regions outside are computed at a lower resolution, sampled such that it can be handled in local memory. We present data structures to compute the set of atoms involved in the update when the *rover* is moved. Fast summation based algo-

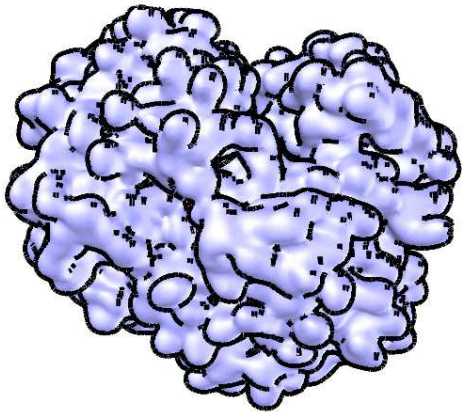
[1]The resolution of the surface is related to the decay rate of the Gaussian function in section §3.3.

rithms are presented to compute the new function dynamically. An adaptive smooth isocontouring technique is introduced to extract smooth meshes of the multiresolution surface for visualization.

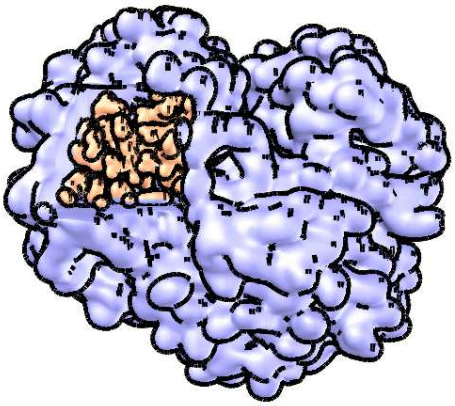
6.3.1 Related Work

There have been various algorithms used in multiresolution modeling in research areas both in and outside of molecular modeling. A good survey of existing techniques are presented in [62].

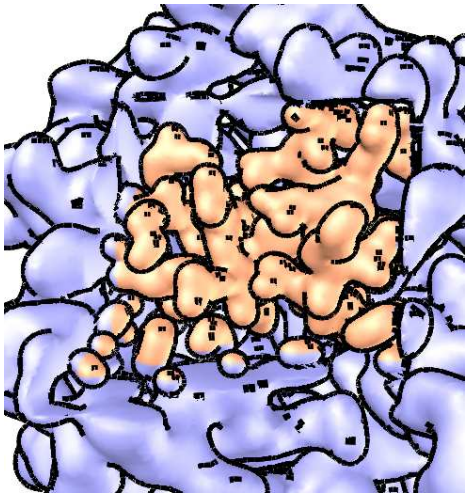
Adaptive isocontouring Marching Cubes (MC) has been a widely used uniform grid isocontouring algorithm [112]. By storing only the cells containing an iso-contour, uniform data grids are converted adaptively into octrees [150]. Several post-processing are performed in which cells meeting certain criteria are merged, producing an adaptive contour. Westermann et al. described an octree-based extraction method [173] that patches the cracks produced by traditional MC by limiting difference of levels between neighboring cells to two and by applying a pyramid averaging scheme to cover the cracks with additional polygons. Dual contouring reported in [87] describes a method that contours on the dual graph of MC, producing crack free isosurfaces.



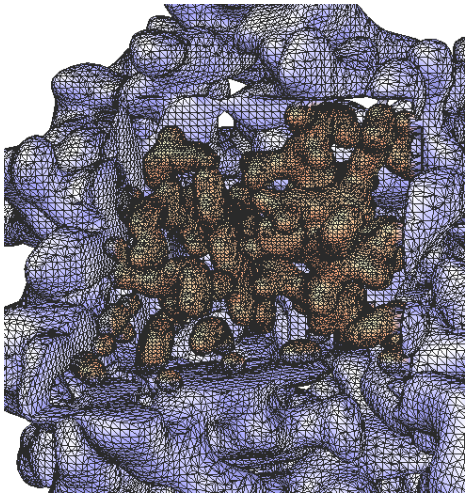
(a) The entire hemoglobin molecule is represented at a coarse resolution, showing the main globular chains.



(b) The focus is moved to a region of interest, which is resolved at a higher resolution.



(c) A zoom into the focus showing smooth adaptive surfaces.



(d) The wireframe rendering shows crack free isocontouring.

Figure 6.6: A single resolution image of the hemoglobin molecule hides important active site details. We provide the user with a *rover* to dynamically compute regions of interest with higher resolution using a combination of novel fast summation algorithms and smooth dual contouring techniques.

6.3.2 Interactive Exploratory Visualization

This program [2]takes an ASCII formatted file containing the center coordinates commonly referred to as the PDB (Protein Data Bank). Each atom's type gives it a unique radius. The user can also select to color the resulting adaptive meshes by properties like the chain number etc. The interface allows the user to change the resolution inside and outside the *rover* and the isovalues. A set of three axis, aligned with the global x, y and z axes (colored red, green and blue for the positive axes) are used to both move and resize the *rover*. A wireframe of a cube is rendered to show the outline of the *rover*. These geometry are rendered with no depth in OpenGL to always keep them on top, we have added depth to the *rover* to show the reader the actual position). Knobs at the end of the axes are used to resize the inner region. The traditional user interface transformations like translation, zoom and rotation are provided.

On loading a PDB file, an adaptive surface is extracted with the *rover* in a default position, the outer region at low resolution 5Å and the inner region at atomic resolution and more densely sampled. UI widgets are provided for the user to change all of the above parameters.

[2]The implementation and all isocontouring details were done by Powei Feng, an undergraduate at The University of Texas at Austin

6.3.3 Multiresolution Molecular Surfaces

Let us consider a protein with M atoms centered at $\mathbf{c}_i$, and radii $r_i, 0 \leq i \leq M$. Since we are dealing with proteins and RNA, we can simplify our problem by assuming that the radii come from a small discrete set, so each set can be computed separately and summed up (the fast summation algorithms do not consider different radii for the kernels). Let the roving cube break this set into two disjoint sets V_{out}, V_{in} with $M_{out}, M_{in} : M_{out}+M_{in} = M$ atoms each. We allow three different kinds of dynamic updates. First, the resolution of the inner and outer functions (f_{out}, f_{in}), controlled by changing the parameters β_{out} and β_{in}. The isovalues in each region (iso_{out}, iso_{in}) can be changed, showing the skin (molecular surface at isovalue 1) and the backbone (regions of higher density). The user is also allowed to roam the dataset, visualizing regions in higher resolution using the *rover*. In figure 6.7, we show the main steps in our algorithm. These interactions require maintaining a set of active atoms in the *rover*, dynamically updating the new function and a smooth adaptive isocontouring algorithm.

6.3.3.1 Atom Set Query

To speed up the query of particles lying in and out of the sub-volume we construct of a range tree on the input centers [95]. A range tree is an $O(\log(M))$ method (for an M atom system) for determining a subset of the input which lies inside any given range. We consider the construction of the range to be pre-processing and is not included in the actual computation.

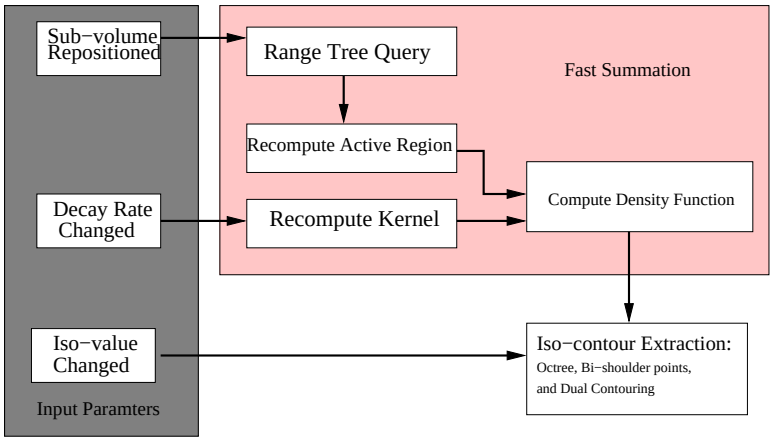


Figure 6.7: A block diagram showing the system implementation.

On every repositioning of the sub-volume, we have to query the range tree to obtain the subset of centers inside the sub-volume and its complement set (centers that are outside the sub-volume.) Note that this is two range queries and not one. The complexity for the range query given a bound is $O(\log(M + k)))$ for some constant k. Range tree also carries the storage overhead of $O(M \log M)$.

6.3.3.2 Fast Density Function Update

For interactive visualization of dynamically updated *rover* and resolution, isovalue parameters, we need algorithms to compute the functions f_{out} and f_{in} efficiently. When the *rover* cube is moved, a new volume f_{in} needs to be computed for isosurfacing. This is probably the most common update operation performed during interaction. We provide precomputation based algorithms to speed up this update.

Direct summation: If we have N output points and M Gaussians, then we can compute the sum at all the points in O(NM) time and O(M+N) memory. If the rate of decay of the Gaussian is high, then we can use a truncated Gaussian and update only around it. Consider a width of w for a truncated Gaussian. Using local summations, we get a computation cost of Mw^3. For grids (where N is large, typically 128^3 to 256^3), and slow decay Gaussians, this operation can be expensive.

Fast summation algorithms The Gaussian function summation can be expressed as a convolution. This allows the functions f_{out}, f_{in} to be quickly computed with a change of β_{out}, β_{in} using the Fourier transform. There are fast Gaussian summation algorithms including general multi-pole methods. We follow the method outlined in section 3.3 to compute both the functions. The cost of updating the function in the *rover* is seen to vary as $\bar{M} \log \bar{M}$ for $\bar{M}$ atoms influencing the inner function. In the case of proteins where a large number of atoms are represented in a smaller subgrid, this dependence on the number of atoms will be a bottleneck to the update process. As before, the summation is approximated as a sum of locally defined functions ϕ such as bsplines or truncated Gaussians (see also [135]): $f(\mathbf{x}) = G \otimes \sum_{i=0}^{M-1} \delta(\mathbf{x} - \mathbf{c}_i) \approx \phi \otimes \sum_{\mathbf{i} \in I_\mathbf{n}} k_{\lfloor \mathbf{x} - \mathbf{i} \rfloor} \delta(\lfloor \mathbf{x} - \mathbf{i} \rfloor)$ where ϕ is a locally supported function, k is a the set of coefficients for ϕ. $I_\mathbf{n}$ is a cubic set of indices: $\{\{i, j, k\} : i, j, k \in -n/2 \,..\, n/2 - 1\}$. Their algorithm uses $O(n^3 \log n + Mm^3 + N^3 \log N)$ time and $O(n^3 + N^3 + M)$ memory, where n is taken to be the same order of M and

controls the error with m for this precomputation. The truncated function has support of size $2m+1$. Using this higher order grid, a simple convolution with ϕ is used to compute f_{in}. Thus, for grids with low atom density, a range tree is used to obtain atoms within the new *rover* and the function computed. For grids with high atom density, we switch to using a precomputed higher order grid. A convolution with ϕ with the correct subset of k gives us f_{in}. This step is independent of $\bar{M}$, but requires the precomputation of the coefficients k.

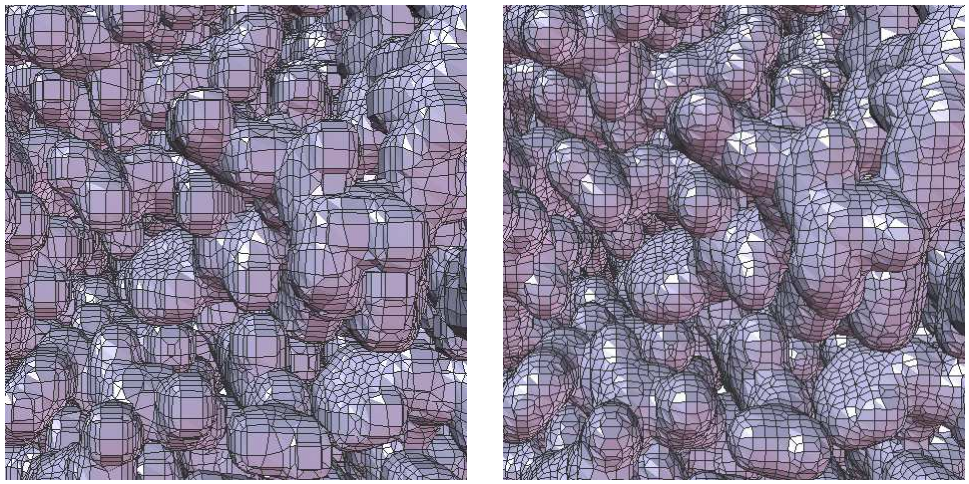
Update β_{out}: The Fourier transform of the new Gaussian can be done analytically. The cost is in multiplying the two set of frequencies ($O(N_{out}^3)$) and performing an inverse Fourier transform ($O(N_{out}^3 \log N_{out})$). We also need to isocontour the outer and boundary regions.

Update β_{in}: For the range tree partitioning method, we obtain the list of atoms within the *rover* in $O(M \log M)$ time and perform a fast summation in approximately $O(M_{in} + N_{in}^3 \log N_{in})$ time. The higher order grid update is more expensive. We would need to precompute the coefficients k again everywhere, and then perform the convolution of ϕ with a subset to update f_{in}.

Rover update: The current β_{in} gives us the width of the truncated kernels. Using the range tree data structure, we query for the set of atoms influencing the *rover* and perform a fast summation to update f_{in}. The mesh needs to be recomputed. If the movement of the rover was small we can reuse the previously compute f_{in} and its corresponding isosurface. The movement of the *rover* also allows us to perform any well known caching algorithm to

store frequently visited active sites in high resolution in our cache.

6.3.3.3 Smooth Adaptive Isocontouring



(a) The traditional QEF minimizer forces sharp edges in the function, leading to unrealistic isosurfaces for the smooth electron density function of molecules.

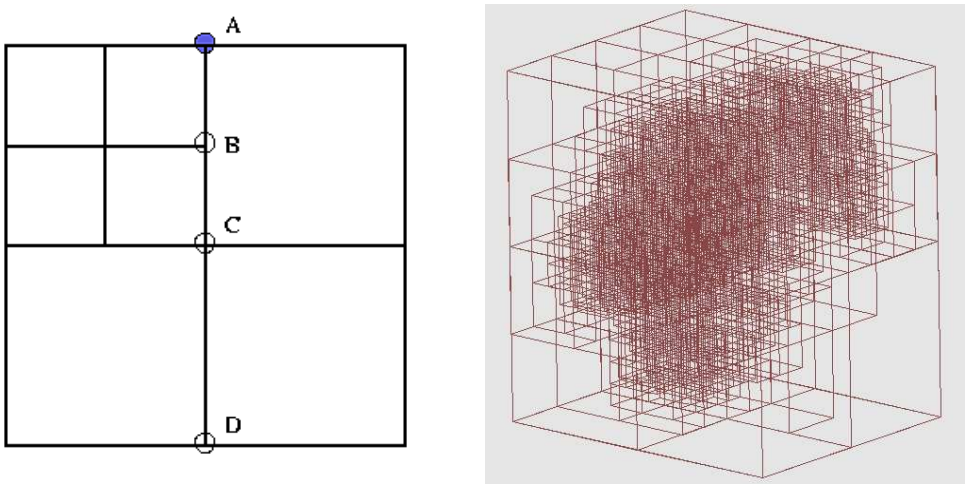
(b) A bishoulder point based dual contouring algorithm results in smooth contours.

Figure 6.8: The hemoglobin molecule (1A00.pdb) is rendered using the traditional QEF minimizer and our bishoulder point based dual contouring algorithm. We see that our method leads to smooth realistic isosurfaces.

We use dual contouring [87] as our surface extraction algorithm. Dual contouring uses the dual map of the primal contouring (Marching Cube) and normal tagging (Hermite Data) to extract the iso-contour. We choose this method primarily because of it avoids the degenerate cases of primary contouring where cracks are introduced. Our algorithm differs in the minimizer computation. Instead of Hermite data and QEF minimizer, we compute a vertex termed *bishoulder point* [111] that closely approximates the true con-

tour. In figure 6.8, we show how the bishoulder point is a better minimizer for smooth functions (If we were to perform isocontouring of objects with sharp edges, we would not use this technique). Lopes and Brodlie [111] detailed in their reports the mathematical reasoning behind bishoulder point's accuracy. We retain the general principle of dual contouring in generating quads. The algorithm traverses each cell recursively. Upon locating a leaf node that contains iso-contour, we check the *sign-change edges* of the cell. Consider the interval defined by the function values of an edge's endpoints. If the iso-value is within that interval, then the edge is a sign-change edge. For example, in Figure 6.9(a), AB, AC, and AD are sign-change edges. We refer to a sign-change edge as being *minimal* when it is an edge of the smallest cell neighboring itself. In Figure 6.9(a), AB is the minimal sign-change edge. If the edge is not minimal, then it is skipped. If the sign-change edge is minimal, then a quad is created, connecting the four bishoulder points of the four cells neighboring the sign-change edge.

Our dual contouring algorithm uses an octree as its surface extraction data structure. The octree is a recursive subdivision of space into variable-sized cells. Its structure is inherently similar to that of a uniform cell division in MC. We can exploit its subdivision structure to produce adaptive cells. This is a simple matter of subdividing cells that borders the sub-volume. Figure 6.9(b) illustrates the use of adaptive cell construction with octree.



(a) (b)

Figure 6.9: (a) is a 2D analog for the sign-changed edge. (b) is an example of adaptive cell construction with octree.

6.3.4 Examples and Timings

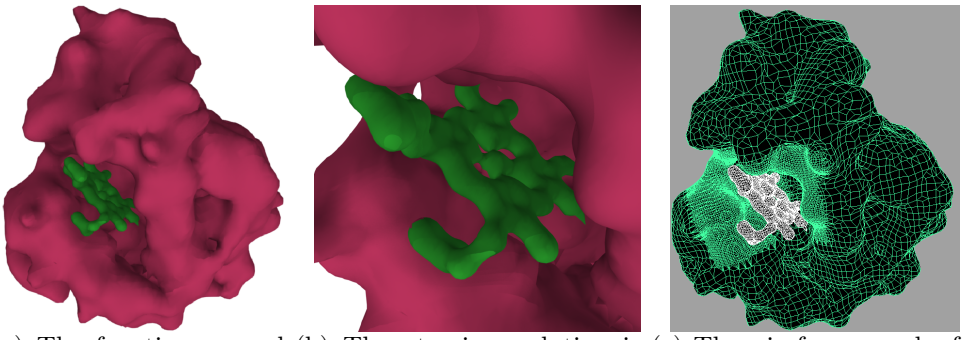
We have implemented the algorithm for the *rover* using standard C++ and QT (www.trolltech.com). All the experiments were done on an AMD Opteron 246. We looked at different size molecules, ranging from the hemoglobin (1A00.pdb) which has approximately 4000 atoms, to the Ribosomal subunits (1J5E.pdb,1JJ2.pdb) which have around 100,000 atoms and the Human Rhinovirus Virus, which has over a million atoms.

In figures 6.10 and 6.11, we show both the methods of multiresolution our algorithm provides. First, the site of interest in the molecule is smoothed using a sharper kernel than the rest of the molecule to differentiate it and provide higher detail in that region. Next, we use our smooth, adaptive dual contouring algorithm to extract an adaptive mesh which provides high detail

at the region of interest and lower resolution elsewhere. These two parameters, the kernel decay rate and the isocontouring mesh refinement can be controlled by the user to obtain feature based functions and visualizations. In figure 6.10, the heme is the active site of the molecule and the region of interest. Hence, the atoms of the heme are smoothed using a gaussian at a 1Åresolution while the rest of the molecule was blurred a coarser resolution of 3Å. Also, by allowing the user to provide a cube around the heme, the adaptive isocontouring algorithm was used to extract a higher resolution mesh around the active site. In figure 6.11, we use the same two multiresolution techniques to show the trimer, a unit of symmetry of the icosahedral Human Rhinovirus. The Large and small Ribosomal subunits are responsible in part for the creation of proteins and widely studied. The main active sites are called as the A,P and E sites. There is also the formation of the cavities and exit tunnels in the bound complex. It is a rich complex with various features including small proteins helping its activity. In figure 6.12, we see that increasing the resolution in the *rover* and changing properties like the grid spacing and color helps the user focus on the regions of interest. The smoothness of the dual contouring is maintained even though there is a sharp increase in both resolution and sampling density. Timing results and the number of quads generated in the adaptive isocontouring are presented in table 6.3.4.

PDB ID	Number of particles	Coarse Volume Summation (secs)	Fine Volume Summation (secs/atoms)	Polyg. Time	Number of Quads
Coarse Volume Resolution: 64^3 — Subvolume Resolution: 52^3					
1A00	4770	0.94	0.32/470	0.72	73181
1J5E	51743	0.94	0.66/10997	0.68	43092
1JJ2	90418	0.93	0.63/10426	0.66	49176
Coarse Volume Resolution: 128 — Subvolume Resolution: 104^3					
1A00	4770	2.95	3.09/474	4.32	313421
1J5E	51743	3.68	2.95/11180	3.82	216307
1JJ2	90418	3.02	2.95/10598	3.54	251904
Coarse Volume Resolution: 256 — Subvolume Resolution: 206^3					
1A00	4770	10.81	25.79/474	28.76	1257644
1J5E	51743	10.89	27.97/11180	29.04	926541
1JJ2	90418	10.35	26.32/10598	23.55	1074166

Table 6.2: This table shows the timing results of our method. All tests are performed on AMD Opteron 246 with 16GB of memory. The subvolume is sampling $(40\%)^3 = 6.4\%$ of the entire input domain. In the third set of results, we perform the surface extraction at very high resolutions, where the small domain in the *rover* is sampled at 206^3.



(a) The function around the heme is extracted at higher resolution.

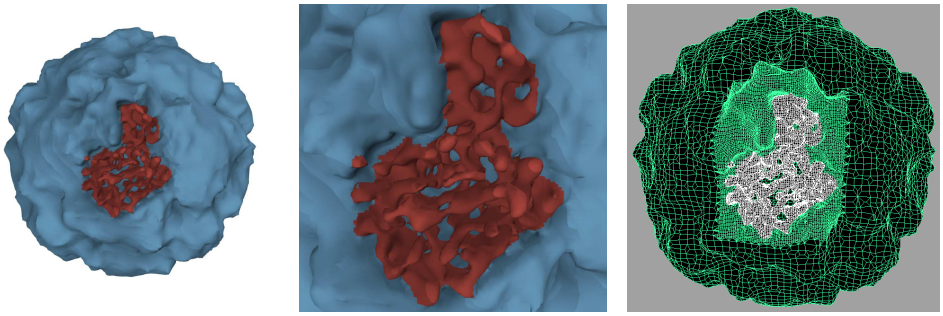
(b) The atomic resolution is able to clearly distinguish the heme atoms.

(c) The wireframe mesh of the isocontour.

Figure 6.10: The myoglobin molecule showing the heme structure. We used a kernel with sharp decay rate, modeling the atomic structure for the heme where the oxy-deoxygenation takes place. A coarser rate of decay was used for the rest of the molecule as the active site is of primary interest for the end user. The region around the heme was extracted at a higher resolution using an adaptive isocontouring algorithm. To maintain the required features, fewer frequencies were required for most of the molecule as compared with the heme.

6.4 Conclusion

The fast summation algorithm and adaptive isocontouring is used to enable interactive exploration of dynamic multiresolution meshes of molecular surfaces with varying decay rates. The *rover* is introduced as a way to explore regions of interest at higher resolutions.

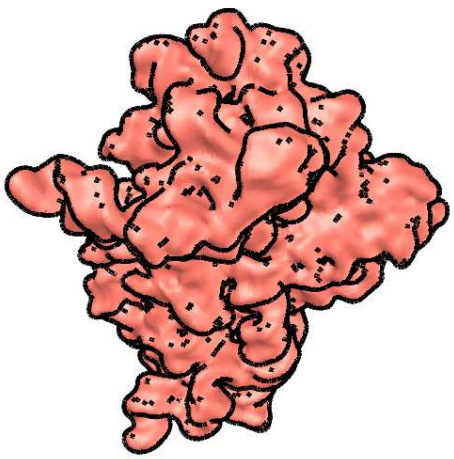


(a) The virus with one capsid shown in higher resolution.

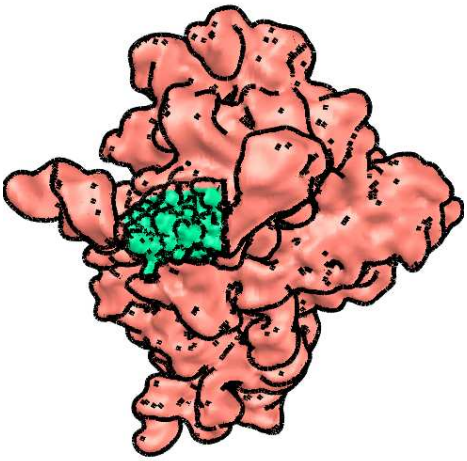
(b) A zoomed view to present the smooth isocontour.

(c) The wireframe mesh of the adaptive isocontour.

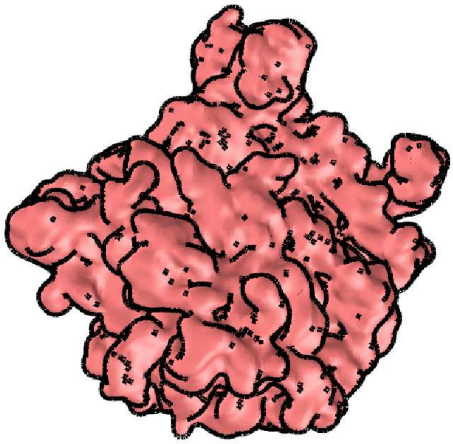
Figure 6.11: The Human Rhinovirus, an icosahedral virus (1FPN.pdb) showing a trimer (a symmetry unit) in higher resolution. The trimer was smoothed using a sharper gaussian than the rest of the virus. Also, an adaptive isocontouring technique was employed to extract a higher resolution mesh in a cube containing the trimer.



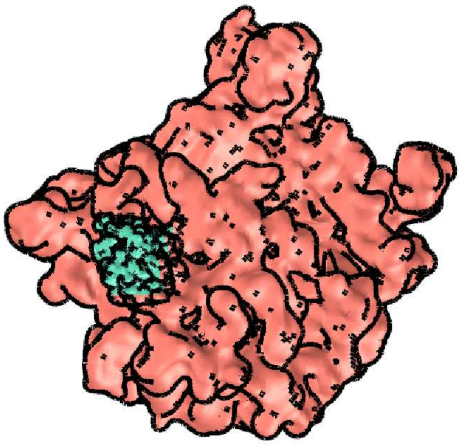
(a) The Small Ribosomal Subunit at a single resolution.



(b) The *rover* is used to visualize a region of interest at higher resolution.



(c) The Large Ribosomal Subunit at a single resolution.



(d) An active site is extracted at higher resolution showing atomic details. The rest of the molecule still presents the global features.

Figure 6.12: We present global and local views in the Ribosomal subunits and contrast it with using a single overall resolution.

Chapter 7

Conclusions

The study of structures and interactions of large biomolecular complexes have led to many challenging research problems in computational geometry, modeling and visualization. These studies lead to an understanding of complex life processes, the working of cells at a molecular level, diseases and drug discovery. We will briefly summarize the results and present some future work.

7.1 Summary of Results

Molecular surfaces The structure of proteins, and more specifically, its interface and interface properties play an important role in its interaction with its environment. Hence fast and accurate evaluation of molecular interfaces and properties is of fundamental interest to those studying biomolecules. As the size of molecules and molecular assemblies being studied increases, we need fast scalable algorithms to compute their interfaces.

Analytic model: Isotropic Gaussian kernels have been traditionally used to describe atoms due to their ability to approximate electron density maps and their analyticity. A suitable level set of electron density describes the molecular

surface. Exponentials have also been considered as suitable kernels to model the interface. We have derived fast summation algorithms to compute this function. For a molecule with M atoms, where the Fourier coefficients of the kernel have a decay of the type $1/\omega^3$, we have developed an $O(M+n^3 \log n+N)$ time, Fourier based algorithm to compute N approximate, irregular samples of a level set surface and its derivatives within a relative L_2 error norm ϵ, where n is $O(M^{1/3}\epsilon^{1/3})$. Specifically, a truncated Gaussian of the form e^{-bx^2} has the above decay, and n grows as $\sqrt{b}$ [154]. We also present an algorithm to update the function when domains of proteins move relative to each other.

Adaptive grid based model: The adaptive grid algorithm computes a family of surfaces for a given molecule, regions of interest and aids in computing areas, volumes and other geometrical properties. Unlike traditional techniques of computing molecular surfaces that involve complex data structures, we show how C_0 smooth approximations with no self intersections can be obtained using a more conventional adaptive grid data structure.

Protein-protein docking The function of proteins and behavior of molecular assemblies in cells is guided by their structural and electrostatic interactions. Protein-ligand docking is used as a first step in screening during drug discovery. Previous grid based solution lack error bounds and complexity analysis. Many spherical harmonics based approaches have high computational and memory costs. Computational geometry based approaches depend on heuristics for correctness. We have developed Fourier series based algo-

rithms which gives users error bounds and provably faster solutions. The key observation in shape based matching algorithm is the formulation of the problem as Fourier series expansion [10]. We reduce the shape based rigid docking problem to maximizing the convolution of four functions over all translations and rotations. Expressing this using Fourier coefficients, we speed up the translational search. To compute the Fourier coefficients of the function of atoms centers, we use a Non-Equispaced Fast Fourier Transform algorithm. The Fourier coefficients of the truncated atomic kernels are done in 1D numerically to a high accuracy as a preprocessing step. This transforms the docking profile into convolution of a uniform grid with a compactly supported smooth function. This non-convex optimization problem of finding peaks in the docking profile is computed with a new adaptive search algorithm. The docking algorithm is shown to scale linearly with the size of the molecules. This scalability makes it suitable for large molecular complexes. The computational cost of these algorithms are an order of magnitude improvement over existing algorithms and are the first to provide error bounds. These suite of algorithms are implemented in my docking software package **$\mathbf{F^2}$ dock** (for Fast and Fourier dock). This research has led to a multidisciplinary NIH grant with The Scripps Research Institute, TSRI, and was joint work done with Dr Castrillon and Dr Olson, Dr Sanner from TSRI.

Flexibility: Flexibility modeling is important in capturing the interaction of flexible proteins. A hierarchy of flexibility (Flexible Chain Complex) is built using normal mode analysis on the proteins. We also provide a file format

for users to specify their own flexibility. This, together with a multiresolution definition of the structure and affinity functions is used to adaptive sample flexibility and orientation space. Finally, a greedy algorithm is used to refit side chains at interfaces. This has being implemented in our software **F^3 dock** (for Fast Flexible and Fourier dock).

Visualization of large molecular models Visualization can be key to observing, understanding and even steering macromolecular interactions. Large molecules like ribosomes, microtubules, viruses etc contain a hundred thousand to nearly tens of million of atoms. Better imaging techniques are bridging the gap between proteins and cells. Interactive visual exploration of such large complex structures provides a wealth of information for molecular biologists. To capture both local and global features of interest, multiresolution techniques are required. We have developed and implemented new algorithms for real-time imposter rendering of large macromolecules, dynamic computation and visualization of multiresolution molecular surfaces, adaptive mesh acceleration data structures, time varying volume visualization and error bounded hierarchical basis feature preserving molecular density compression.

Imposter based visualization: A traditional molecule visualization paradigm is explicit representation of atoms, bonds and secondary structures. Large biomolecular complexes are generally in the order of millions of atoms, each represented as a sphere, with bonds shown as cylinders. Groups of atoms called secondary structure motifs are shown as helices and sheets as these are

commonly occurring structures in proteins. While molecular visualization software has advanced over the years, today, most tools still operate on individual molecular structures with limited facility to manipulate large multi-component complexes. They cannot handle the triangle soup generated from these complexes. We approach this problem by extending 3D image-based rendering via programmable graphics units, resulting in an order of magnitude speedup over traditional triangle-based rendering. We developed a new technique to render well defined curved surfaces like spheres, cylinders and helices using single quads on programmable graphics cards [13]. The symmetry of the objects enables us to capture shape, normal and depth information through the transformation pipeline in a single dimension. Hierarchical clustering of atoms aids in level of detail rendering using imposters, further improving performance.

Dynamic multiresolution molecular surfaces: As the complexity of the molecules increase, it becomes important to present both local features like active sites and global features like the interaction of tertiary motifs and the overall quaternary structure. We have developed a dynamic surface generation and exploratory visualization technique that allows the user to explore these biomolecular complexes, at interactive speeds, with regions of interest captured via a roaming microscopic cube. The molecular surface of these biomolecular structures is defined as a level set of a function that is dynamically constructed by the summation of decaying atomic electron density kernels. We compute adaptive resolution molecular surfaces by allowing the users to control both the grid resolution and kernel decay rates in the regions of interest. To achieve

interactivity, we utilize tree based data structures for atomic center identification within a roaming microscopic cube and fast local and global update algorithms coupled to adaptive crack-free contouring. Our dynamic multiresolution surface generation algorithm is shown to be scalable to very large complex biomolecules.

I have implemented these visualization techniques in a high-performance, interactive molecular exploration tool we call **TexMol**, short for Texture Molecular viewer.

7.2 Future Work

Multiscale biomolecular modeling and visualization is currently possible with the right collaborative effort. There has been a deluge of molecular structures and related data in recent years. From the imaging community, we are obtaining higher resolution models of neuromuscular junctions and cells, far larger than the proteins people have been dealing with. Molecular biologists have been analyzing the structure and functions of these macromolecular assemblies. Computer scientists are knowledgable in systems modeling, visualization techniques, algorithm design and computational techniques. The convergence of these methods can lead to a better understanding of the complex processes in a living organism, with immediate consequences in treating diseases. We have worked on some steps in achieving physiologically correct models, and many interesting research questions remain.

Flexible models: Proteins are known to be flexible and this property plays an important role in its interactions. Flexible protein-protein docking greatly increases the degrees of freedom. But Normal Mode Analysis gives us a parameterization for atom positions. The sine basis functions can be factored into the Fourier based docking equation from [10] to give a compact mathematical representation for the flexible docking problem in a single equation. I believe that solutions to this equation would yield better docking profiles than existing algorithms. A fast and accurate flexible docking algorithm would make many interesting problems in computational biology tractable.

Multiscale modeling: Functions of proteins are tightly coupled to their structure, often resembling man-made mechanical devices. This enables a robust specification of the protein, describing and linking its structural components and behavior. This leads to natural adaptive descriptions of larger macromolecular assemblies. The scale of the models, ranging from microns to angstroms, and milliseconds to nanoseconds requires adaptive and multiscale modeling and visualization algorithms. I would like to investigate both the formal specification of protein structure and function, and adaptive algorithms for modeling, computation and visualization of important processes. I believe that a powerful yet simple specification language will help improve collaborations in this field.

Computational virtual screening: This is an exciting area of research, where protein-ligand docking is applied in a high throughput pipeline to match candidate drugs and disease causing agents. Our flexible model and fast docking

is currently being tested on databases of small ligands to see how they interact with fragments of viruses.

There are also a variety of other problems in Computational Biology that I have worked on in joint projects, including: Geometry based classification and analysis of proteins, complementary space analysis, evolutionary methods to understand and improve function of proteins, and correlations between charge and curvature of protein interfaces. I have also dabbled in other areas including surface and fat surface modeling using A-patches and information visualization.

In summary, I have developed novel techniques to efficiently handle protein-protein interactions, compute multiresolution surfaces and properties and visualize large molecular domains. These are a few of the challenges that hold the key to understanding complex life processes, the nature of diseases and drug design. In future, I would like to work on enabling large multiscale biomolecular modeling to construct reliable and physiologically correct models. I believe that this is a research area rich with problems that I am interested in and one that holds promise to a lot of exciting discoveries.

Appendices

Appendix A

Error Bounds

In this section, we derive values for n, m, α to keep the theoretical error within the user defined error threshold ϵ. This analysis thus provides us with both a theoretical an upper bound on the computational and storage costs of the algorithm and variables. We consider the Gaussian kernel.

Simple properties of the Gaussian function Let us define the 1D Gaussian function as:

$$g(x - x_c) = e^{\beta/r^2(x-x_c)^2}$$

Let $\mathbf{x} = \{x, y, z\},\ \mathbf{x}_c = \{x_c, y_c, z_c\}$. Then the 3D Gaussian kernel:

$$\begin{aligned} g(\mathbf{x} - \mathbf{x_c}) &= e^{\beta(\frac{|\mathbf{x}-\mathbf{x}_c|^2}{r^2}-1)} \\ &= e^{-\beta}e^{\beta/r^2(|\mathbf{x}-\mathbf{x}_c|^2)} \\ &= e^{-\beta}e^{\beta/r^2(x-x_c)^2}e^{\beta/r^2(y-y_c)^2}e^{\beta/r^2(z-z_c)^2} \\ &= e^{-\beta}g(x - x_c)g(y - y_c)g(z - z_c) \end{aligned}$$

- The Gaussian is radially symmetric.
- It is a tensor product of 1D Gaussians.

Fourier coefficients of a truncated Gaussian The Fourier transform of a Gaussian is well known.

$$G(\mathbf{k}) = e^{-\beta} G(k_1)G(k_2)G(k_3),\ G(e^{\beta/r^2x^2})(k) = \sqrt{\frac{-\pi r^2}{\beta}} e^{\pi^2 r^2 k^2/\beta}$$

We are concerned with obtaining the Fourier series coefficients of the truncated Gaussian. Thus in each dimension, we need the fourier coefficients of

$$g_{trunc}(x) = g(x).b(x),\ b(x) = \begin{cases} 1 & -0.5 \leq x < 0.5, \\ 0 & elsewhere \end{cases}$$

For a given β and r, we can compute the Fourier series of the above numerically in MAPLE, or we can use the approximation provided by a large 1D FFT. The above can also be written as the convolution of a sinc and a Gaussian in the Fourier domain. We use this to find the error bounds of the algorithm. Since we are interested in particular values for the decay rate parameter in most applications, this step can be a precomputation.

A.1 Approximations in Fast Summation Algorithm

Let us denote by $f, \hat{f}$ the exact and computed summation. Let the exact and approximate Fourier series coefficients of the function of atom centers and the truncated Gaussian be $C_{\boldsymbol{\omega}}, \hat{C}_{\boldsymbol{\omega}}, G_{\boldsymbol{\omega}}, \hat{G}_{\boldsymbol{\omega}}$.

$$
\begin{aligned}
||\hat{f} - f||_2 &= || \sum_{\boldsymbol{\omega} \in I_\infty} C_{\boldsymbol{\omega}} G_{\boldsymbol{\omega}} e^{2\pi i \boldsymbol{\omega}.\mathbf{x}} - \sum_{\boldsymbol{\omega} \in I_n} \hat{C}_{\boldsymbol{\omega}} \hat{G}_{\boldsymbol{\omega}} e^{2\pi i \boldsymbol{\omega}.\mathbf{x}} + \epsilon_1 ||_2 \\
&= || \sum_{\boldsymbol{\omega} \in I_n} (C_{\boldsymbol{\omega}} G_{\boldsymbol{\omega}} - \hat{C}_{\boldsymbol{\omega}} \hat{G}_{\boldsymbol{\omega}}) e^{2\pi i \boldsymbol{\omega}.\vec{x}} + \sum_{\boldsymbol{\omega} \in I_\infty \setminus I_n} C_{\boldsymbol{\omega}} G_{\vec{k}} e^{2\pi i \boldsymbol{\omega}.\vec{x}} + \epsilon_1 ||_2 \\
&\leq || \sum_{\boldsymbol{\omega} \in I_n} (C_{\boldsymbol{\omega}} G_{\vec{k}} - \hat{C}_{\boldsymbol{\omega}} \hat{G}_{\boldsymbol{\omega}}) e^{2\pi i \boldsymbol{\omega}.\vec{x}} ||_2 + || \sum_{\boldsymbol{\omega} \in I_\infty \setminus I_n} C_{\boldsymbol{\omega}} G_{\boldsymbol{\omega}} e^{2\pi i \boldsymbol{\omega}.\vec{x}} ||_2 + ||\epsilon_1||_2 \\
&= \sqrt{(\sum_{\boldsymbol{\omega} \in I_n} (C_{\boldsymbol{\omega}} G_{\boldsymbol{\omega}} - \hat{C}_{\boldsymbol{\omega}} \hat{G}_{\boldsymbol{\omega}})^2)} + \sqrt{(\sum_{\boldsymbol{\omega} \in I_\infty \setminus I_n} (C_{\boldsymbol{\omega}} G_{\boldsymbol{\omega}})^2)} + ||\epsilon_1||_2 \\
&= e_1 + e_2 + e_3
\end{aligned}
$$

e_1 and e_3 are errors due to NFFT', NFFT. We choose m, α such that these errors are both less than one third the user allowed error. We are left to choose n such that the second term is also less than $\epsilon/3$. For error bounds for m, α, please look at [136]. In practise, $\alpha \approx 2, m \approx 3$ gives us errors less than 1% which is sufficient for our applications.

Decay of smooth kernel Fourier coefficients Now we present the decay of the Gaussian $G_{\boldsymbol{\omega}}$ and show that it is upper bounded by $1/\boldsymbol{\omega}$

The truncated gaussian is the product of a gaussian and a box function. In the fourier domain, this can be expressed as a convolution between a sync

and a gaussian. the sync itself can be upper bounded by a function of type $1/x$. Since the second function goes to infinity at 0, we replace the $1/x$ with a flat function near the origin. Specifically,

$$FT(Box) = \frac{2\sin(2\pi t)}{2\pi t} \leq \begin{cases} 2 & -1 \leq t \leq 1 \\ \frac{1}{\pi t} & t > 1 \\ -\frac{1}{\pi t} & t < 1 \end{cases}$$

Also let

$$\hat{K}(t) = FT(K)(t) = \sqrt{\frac{\pi}{B}} e^{\frac{-\pi^2 t^2}{B}}$$

thus we get $G_\omega, \omega > 1$ as

$$G_\omega = \int_{-\infty}^{\omega-1} \frac{\hat{K}(t)}{\pi(\omega - t)} dt + \int_{\omega-1}^{\omega+1} 2\hat{K}(t) dt + \int_{\omega+1}^{\infty} \frac{\hat{K}(t)}{\pi(t - \omega)} dt$$

We show that this function has a decay of $O(1/\omega)$. Also, due to symmetry, $G_\omega = G_{-\omega}$.

$\mathbf{I_1}$:

Let

$$I_1 = \int_{-\infty}^{0} \frac{\hat{K}(t)}{\pi(\omega - t)} dt, I_2 = \int_{0}^{\omega-1} \frac{\hat{K}(t)}{\pi(\omega - t)} dt, I_3 = \int_{\omega-1}^{\omega+1} 2\hat{K}(t) dt, I_4 = \int_{\omega+1}^{\infty} \frac{\hat{K}(t)}{\pi(t - \omega)} dt$$

$$I_1 = \int_{-\infty}^{0} \frac{\hat{K}(t)}{\pi(\omega - t)} dt \leq \int_{-\infty}^{0} \frac{\hat{K}(t)}{\pi\omega} dt$$

Since

$$\int_{-\infty}^{0} \hat{K}(t) dt = \int_{-\infty}^{0} \sqrt{\frac{\pi}{B}} e^{\frac{-\pi^2 t^2}{B}} dt = \frac{1}{2\sqrt{\pi}}$$

$$I_1 \leq \frac{1}{2\pi\sqrt{\pi}} \frac{1}{\omega}$$

$\mathbf{I_2}$:

$$I_2 = \int_{0}^{\omega-1} \frac{\hat{K}}{\pi(\omega - t)} dt = \int_{0}^{1} \frac{\hat{K}}{\pi(\omega - t)} dt + \int_{1}^{\omega-1} \frac{\hat{K}}{\pi(\omega - t)} dt = I_{2,1} + I_{2,2}$$

Bounding $I_{2,1}$:

$$I_{2,1} = \int_{0}^{1} \frac{\hat{K}}{\pi(\omega - t)} dt \leq \int_{0}^{1} \frac{\hat{K}}{\pi\omega} dt = \frac{1}{\omega\sqrt{\pi B}} \int_{0}^{1} e^{-\pi^2 t^2 / B} dt$$

Let $y = \pi t/\sqrt{B}$. Hence, $y^2 = \pi^2 t^2/B$, $dt = \sqrt{B}/\pi dy$, $t = 0 \rightarrow y = 0$ and $t = 1 \rightarrow y = \pi/\sqrt{B}$.

therefore,

$$I_{2,1} \leq \frac{1}{\omega\pi\sqrt{\pi}} \int_0^{\pi/\sqrt{B}} e^{-y^2} dy = \frac{1}{2\pi\omega}(\frac{2}{\sqrt{\pi}}) \int_0^{\pi/\sqrt{B}} e^{-y^2} dy = \frac{1}{2\pi\omega} erf(\frac{\pi}{\sqrt{B}})$$

Bounding $I_{2,2}$:

Let

$$f_1 = \sqrt{\frac{\pi}{B}} e^{-\pi^2 t^2/B},\ f_2 = \frac{c}{t^2}$$

$$Roots(f_1 - f_2) = (\frac{\sqrt{-LW(\frac{-\pi^2 c}{\sqrt{\pi B}})B}}{\pi}, \frac{\sqrt{LW(\frac{-\pi^2 c}{\sqrt{\pi B}})B}}{\pi})$$

We used Maple to compute the roots. LW stands for the LambertW function. Since roots of complex numbers are complex, and using the properties of the LambertW function, there are no real roots (letting f_2 bound f_1) if:

$$c \geq \frac{\sqrt{B}}{e\pi^{3/2}}$$

$$\int_1^{\omega-1} \frac{1}{\pi(\omega - t)} \frac{1}{t^2} dt = \frac{1}{\omega}(\frac{\log(\omega-1)}{\omega} - \frac{1}{\omega-1} - \frac{\log(2\omega-1)}{\omega} + 1 + \frac{\log(\omega+1)}{\omega}) \leq \frac{3}{\omega}$$

$$I_{2,2} \leq 3\frac{\sqrt{B}}{e\pi^{3/2}}\frac{1}{\omega}$$

$\mathbf{I_3}$:

$$I_3 = \int_{\omega-1}^{\omega+1} 2\hat{K}dt \leq 4\sqrt{\frac{\pi}{B}}e^{-\pi^2(\omega-1)^2/B}$$

Let

$$I_3 \leq \frac{c}{\omega},\ \hat{\omega} = \omega - 1,\ f_1 = \frac{d}{\hat{\omega}},\ f_2 = 4\sqrt{\frac{\pi}{B}}e^{-\pi^2\hat{\omega}^2/B}$$

$$Roots(f_1 - f_2) = \frac{d}{\sqrt{\pi/B}}\sqrt{\frac{-LW(-1/8\pi d^2)}{2\pi d^2}}$$

there is no intersection when there is no real root. this implies that f_1 is an upper bound as required. For no real roots we get:

$$d \geq \sqrt{\frac{8}{\pi e}} \rightarrow c > 4\sqrt{2\pi e}$$

$$I_3 \leq 4\sqrt{\frac{2}{\pi e}}\frac{1}{\omega}$$

$\mathbf{I_4}$:

$$I_4 = \int_{\omega+1}^{\infty} \frac{\hat{K}}{\pi(t-\omega)} dt \leq \int_{\omega+1}^{\infty} \frac{\hat{K}}{\pi} dt \leq \int_{\omega+1}^{\infty} \frac{\pi}{B} \frac{e^{-\pi^2 t/B}}{\pi} dt = \frac{e^{-\pi^2(\omega+1)/B}}{\pi^2}$$

Let

$$I_4 \leq f_1 = \frac{c}{\omega},\ f_2 = \frac{e^{-\pi^2(\omega+1)/B}}{\pi^2}$$

$$Roots(f_1 - f_2) = \frac{-B\ LW(\frac{-pi^4 c e^{\pi^2/B}}{B})}{\pi^2}$$

there is no intersection when there is no real root. this implies that f_1 is an upper bound as required. For no real roots we get:

$$\frac{-\pi^2 c e^{\pi^2/B}}{B} < -e^{-1} \rightarrow I_4 \leq \frac{B e^{-(1+\pi^2/B)}}{\pi^4} \frac{1}{\omega}$$

$$G_\omega \leq max(\frac{1}{2\pi\sqrt{\pi}}, \frac{1}{2\pi} erf(\frac{\pi}{\sqrt{B}}), \frac{3\sqrt{B}}{e\pi^{3/2}}, 4\sqrt{\frac{2}{\pi e}}, \frac{B e^{-(1+\pi^2/B)}}{\pi^4}) \frac{1}{\omega}, \quad (\omega \geq 2)$$

For our applications, for all practical values of B:

$$G_\omega \leq 4\sqrt{\frac{2}{\pi e}}\ \frac{1}{\omega} \tag{A.1}$$

A.2 Relation between Number of Fourier Coefficients and Error

We need to find the minimum value for n which satisfies

$$\frac{||\hat{f} - f||_2}{||f||_2} \leq \frac{\epsilon}{3}$$

We first bound the denominator as follows:

$$\begin{aligned}
||f||_2 &= \sqrt{\int_{I_1} (\sum_{j=1}^{M} c_j G(\mathbf{x} - \mathbf{x}_j))^2 d\mathbf{x}} \\
&\geq \sqrt{\int_{I_1} \sum_{j=1}^{M} (c_j)^2 (G(\mathbf{x} - \mathbf{x}_j))^2 d\mathbf{x}} \\
&\geq \sqrt{min_j(|c_j|)^2 \int_{I_1} \sum_{j=1}^{M} (G(\mathbf{x} - \mathbf{x}_j))^2 d\mathbf{x}} \\
&= \sqrt{M min_j(|c_j|)^2 V}, \; V = \int_{I_1} G(\mathbf{x})^2 d\mathbf{x}
\end{aligned}$$

In practise, we can find n as follows:

$$|C_{\boldsymbol{\omega}}| \leq \sum_{j=1}^{M} |c_j||e^{-2\pi i \mathbf{x}_j . \boldsymbol{\omega}}| = \sum_{j=1}^{M} |c_j| = ||c||_1$$

$$\sum_{\boldsymbol{\omega}\in I_\infty\setminus I_n}(C_{\boldsymbol{\omega}}G_{\boldsymbol{\omega}})^2 \leq max|C_{\boldsymbol{\omega}}|^2 \sum_{\boldsymbol{\omega}\in I_\infty\setminus I_n}(G_{\boldsymbol{\omega}})^2 = (||c||_1)^2 \sum_{\boldsymbol{\omega}\in I_\infty\setminus I_n}(G_{\boldsymbol{\omega}})^2$$

Let $D = M\ min_j(|c_j|^2)C$, error due to term $e_2 = \epsilon/3$

$$\begin{aligned}
(||c||_1)^2 \sum_{\boldsymbol{\omega}\in I_\infty\setminus I_n}(G_{\boldsymbol{\omega}})^2 &\leq D(\frac{\epsilon}{3})^2 \\
(||c||_1)^2(\sum_{\boldsymbol{\omega}\in I_\infty}(G_{\boldsymbol{\omega}})^2 - \sum_{\boldsymbol{\omega}\in I_n}(G_{\boldsymbol{\omega}})^2) &\leq D(\frac{\epsilon}{3})^2 \\
(||c||_1)^2(\frac{C}{2\pi} - \sum_{\boldsymbol{\omega}\in I_n}(G_{\boldsymbol{\omega}})^2) &\leq D(\frac{\epsilon}{3})^2 \\
\sum_{\boldsymbol{\omega}\in I_n}(G_{\boldsymbol{\omega}})^2 &\geq \frac{C}{2\pi} - \frac{D}{(||c||_1)^2}(\frac{\epsilon}{3})^2
\end{aligned}$$

Therefore: $n = min(\hat{n}) : \sum_{\boldsymbol{\omega}\in I_{\hat{n}}}(G_{\boldsymbol{\omega}})^2 \geq \frac{C}{2\pi} - \frac{D}{(||c||_1)^2}(\frac{\epsilon}{3})^2$.

We can also obtain the following relation which shows the dependence of n on the input variables.

$$\begin{aligned}
\frac{\sqrt{\sum_{\boldsymbol{\omega}\in I_\infty\setminus I_n}(C_{\boldsymbol{\omega}}G_{\boldsymbol{\omega}})^2}}{||f||_2} &\leq \frac{\epsilon}{3} \\
\sum_{\boldsymbol{\omega}\in I_\infty\setminus I_n}(C_{\boldsymbol{\omega}}G_{\boldsymbol{\omega}})^2 &\leq (\frac{\epsilon||f||_2}{3})^2 \\
\sum_{\boldsymbol{\omega}\in I_\infty\setminus I_n}(G_{\boldsymbol{\omega}})^2 &\leq (\frac{\epsilon||f||_2}{3||c||_1})^2 \\
2\sum_{\omega=n/2+1}^{\infty}(G_\omega)^2 + (G_{n/2})^2 &\leq (\frac{\epsilon||f||_2}{3||c||_1})^{2/3}
\end{aligned}$$

From the decay of G_ω, we have:

$$2\sum_{\omega=n/2+1}^{\infty}\frac{1}{(G_\omega)^2}+\frac{4}{n^2}\leq\frac{\pi^2e^2}{32}(\frac{\epsilon||f||_2}{3||c||_1})^{2/3}$$

Using $\sum_{\omega=n/2+1}^{\infty}1/(G_\omega)^2\leq\int_{\omega=n/2}^{\infty}1/(G_\omega)^2=2/n$,

$$\frac{1}{n}+\frac{1}{n^2}\leq\frac{\pi^2e^2}{128}(\frac{\epsilon||f||_2}{3||c||_1})^{2/3}$$

Hence, we would like to obtain the minimum n which satisfies the above inequality. Since $||c||_1$ is proportional to M and $||f||_2$ is proportional to $\sqrt{M}, (erf(\sqrt{b}/2)/\sqrt{b})^3$, we see that n^3 is proportional to M as n grows. In general, the above is not a tight bound. We use $n^3=M$ and perform a binary search to obtain the desired accuracy.

Appendix B

Software

All the important algorithms in the thesis are implemented in a software package called TexMol. It is written in a combination of C++ and QT. It has been tested on common platforms including Linux, Solaris, Windows and MacOS. TexMol is available for download off our lab's website: http://www.ices.utexas.edu/ccv. The docking software F^2Dock, F^3Dock are being packaged as part of TexMol. The entire software is over 150,000 LOC.

B.1 TexMol

We present only the main features of the software. Please refer to its manual, which is available with the software, for details. The main functionalities can be divided into two broad categories:

- **Visualization algorithms:** These include imposter rendering, surface visualization and volume rendering techniques.
- **General computational algorithms:** Density computations, curvature estimations and construction of molecular surfaces are some of the computational algorithms built into TexMol.

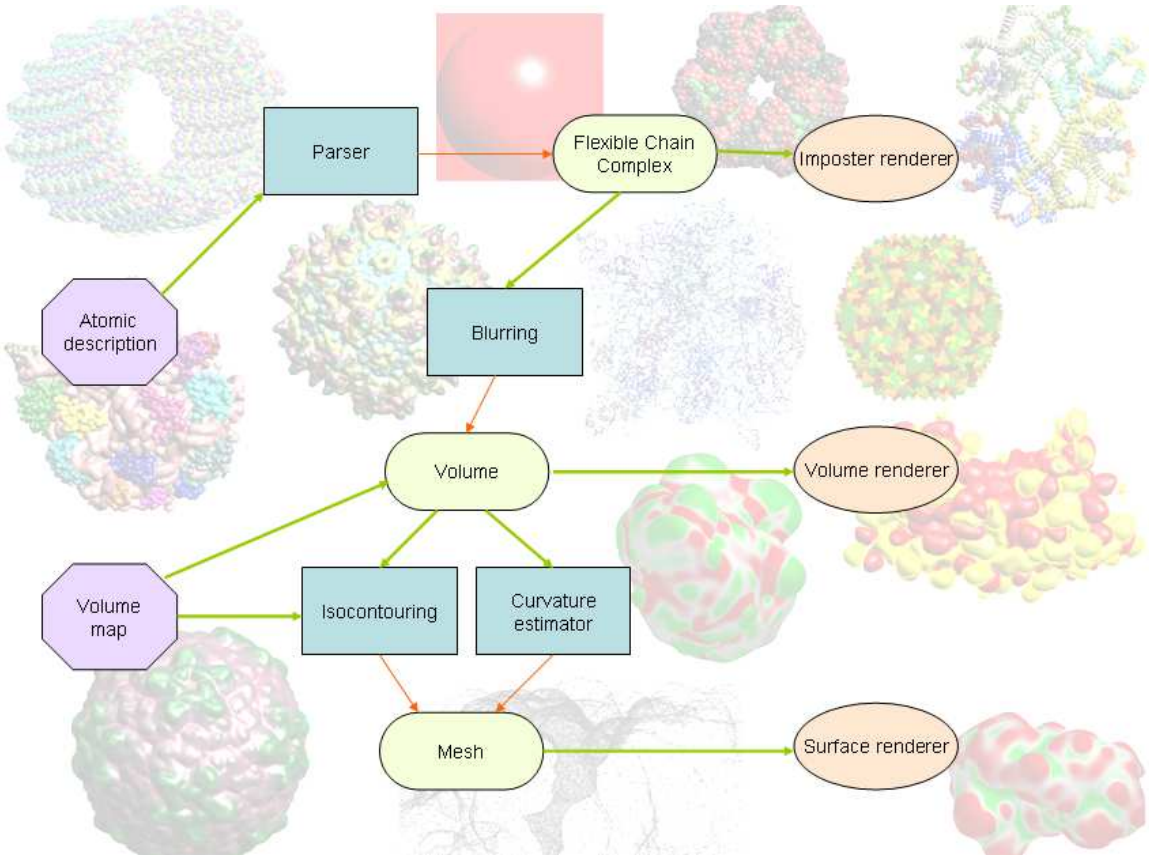


Figure B.1: Visualization modules in TexMol

- **Docking functions:** All functions related to protein-protein docking.

B.1.1 Visualization Algorithms

In figure B.1, we show the main visualization features in TexMol. Some common functions are listed below.

- Imposter based rendering of spheres, cylinders, helices, functions on each

primitive.

- Surface visualization with both wireframe and smooth shading. A telescoping rover is also provided, linked to a fast summation algorithm to view multiresolution images of large molecules.

- Hardware accelerated 3D texture based volume rendering and splatting based volume rendering algorithms. A rudimentary ray tracer is also being incorporated.

B.1.2 General Computational Algorithms

Computational algorithms involve computing molecular structure, properties and functions.

- Fast summation based computation of radial basis functions, with properties represented using different colors and different resolutions.

- Curvatures (both mean and Gaussian) can be computed on surfaces defined as contours of a sum of Gaussians function.

- Hydrophobicity and electrostatics of molecules can be computed using simple sum of kernel definitions.

- Regular and adaptive isocontouring of volumetric data.

B.1.3 Docking Modules

Both F^2Dock, F^3Dock are now available as part of TexMol. They include:

- Construction of surface and skin layers for the protein and ligand.
- Rigid protein-protein docking at multiple resolutions.
- Construction of a Flexible Chain Complex with prioritized flexibilities for adaptive conformational sampling.
- Flexible protein-protein docking modules, including a greedy side chain refit.
- Modules to collect statistics of the docking algorithms.

TexMol also incorporates libraries built over the last 8 or so years at our lab by various students.

Bibliography

[1] Nataraj Akkiraju and Herbert Edelsbrunner. Triangulating the surface of a molecule. *Discrete Applied Mathematics*, 71(1-3):5–22, 1996.

[2] Nataraj Akkiraju, Herbert Edelsbrunner, Ping Fu, and Jiang Qian. Viewing geometric protein structures from inside a cave. *IEEE Computer Graphics and Applications*, 16(4):58–61, July 1996.

[3] Daniel Aliaga, Jon Cohen, Andrew Wilson, Eric Baker, Hansong Zhang, Carl Erikson, Kenny Hoff, Tom Hudson, Wolfgang Stuerzlinger, Rui Bastos, Mary Whitton, Fred Brooks, and Dinesh Manocha. Mmr: an interactive massive model rendering system using geometric and image-based acceleration. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 199–206. ACM Press, 1999.

[4] Ernst Althaus, Oliver Kohlbacher, Hans-Peter Lenhof, and Peter Mller. A combinatorial approach to protein docking with flexible side chains. *Journal of Computational Biology*, 9(4):597–612, August 2002.

[5] Abraham Anderson and Zhiping Weng. Vrdd: applying virtual reality visualization to protein docking and design. *Journal of Molecular Graphics and Modelling*, 17(3-4):180–186, 1999.

[6] Joannis Apostolakis, Andreas Plückthun, and Amedeo Caflisch. Docking small ligands in flexible binding sites. *Journal of Computational Chemistry*, 19(1):21–37, 1998.

[7] Jim Arvo. *Graphics gems*, chapter A simple method for box-sphere intersection testing, pages 335–339. Academic Press Professional, Inc., San Diego, CA, USA, 1990.

[8] David J. Bacon and John Moult. Docking by least-squares fitting of molecular surface patterns. *Journal of Molecular Biology*, 225(3):849–858, June 1992.

[9] Chandrajit Bajaj, Fausto Bernardini, and Kokichi Sugihara. A geometric approach to molecular docking and similarity. Technical Report CSD-TR-94-017, Computer Sciences, Purdue University, March 1994.

[10] Chandrajit Bajaj, Julio Castrillon Candas, and Vinay Siddavanahalli. F2dock: A fast and fourier based error-bounded approach to protein-protein docking. CS Technical report TR-06-57, The University of Texas at Austin, Austin, TX, USA 78712., November 2006.

[11] Chandrajit Bajaj, Julio E. Castrillon-Candas, Vinay Siddavanahalli, and Zaiqing Xu. Hierarchical compressed volumetric representations of molecular structures. ICES Report 04-59, The University of Texas at Austin, Austin, TX, USA 78712, December 2004.

[12] Chandrajit Bajaj, Julio E. Castrillon-Candas, Vinay Siddavanahalli, and Zaiqing Xu. Compressed representations of macromolecular structures and properties. *Structure*, 13(3):463–471, March 2005.

[13] Chandrajit Bajaj, Peter Djeu, Vinay Siddavanahalli, and Anthony Thane. Texmol: Interactive visual exploration of large flexible multi-component molecular complexes. *Proceedings of the IEEE Visualization*, pages 243–250, 2004.

[14] Chandrajit Bajaj, H. Y. Lee, R. Merkert, and Valerio Pascucci. Nurbs based b-rep models for macromolecules and their properties. In *Proceedings of the fourth ACM symposium on Solid modeling and applications*, pages 217–228. ACM Press, 1997.

[15] Chandrajit Bajaj, Valerio Pascucci, Arial Shamir, Robert J. Holt, and Arun N. Netravali. Multiresolution molecular shapes. Technical report, TICAM, Univ. of Texas at Austin, Dec. 1999.

[16] Chandrajit Bajaj, Valerio Pascucci, Arial Shamir, Robert J. Holt, and Arun N. Netravali. Dynamic maintenance and visualization of molecular surfaces. *Discrete Applied Mathematics*, 127(1):23–51, 2003.

[17] Chandrajit Bajaj and Vinay Siddavanahalli. Fast error-bounded surfaces and derivatives computation for volumetric particle data. ICES Report 06-03, The University of Texas at Austin, Austin, TX, USA 78712, January 2006.

[18] Chandrajit Bajaj, Guoliang Xu, Robert J. Holt, and Arun N. Netravali. Nurbs approximation of a-splines and a-patches. *International Journal of Computational Geometry and Applications*, 13(5):359–389, November 2003.

[19] Nathan A. Baker, David Sept, Simpson Joseph, Michael J. Holst, and J. Andrew McCammon. Electrostatics of nanosystems: application to microtubules and the ribosome. *Proceedings of the National Academy of Sciences, USA*, 98(18):10037–10041, August 2001.

[20] Karine Bastard, Aurelien Thureau, Richard Lavery, and Chantal Prevost. Docking macromolecules with flexible segments. *Journal of Computational Chemistry*, 24(15):1910–1920, November 2003.

[21] Helen M. Berman, John Westbrook, Zukang Feng, Gary Gilliland1, T. N. Bhat, Helge Weissig, Ilya N. Shindyalov, and Philip E. Bourne. The protein data bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

[22] James F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982.

[23] Nathalie S. Boutonnet, Marianne J. Rooman, and Shoshana J. Wodak. Automatic analysis of protein conformational changes of by multiple linkage clustering. *Journal of Molecular Biology*, 253(4):633–647, 1995.

[24] Samuel F. Boys. Electronic wave functions, i: A general method of calculation for the stationary states of any molecular system. *Proceedings*

of the Royal Society of London. Series A, Mathematical and Physical Sciences, 200(1063):542–554, February 1950.

[25] Bernard R. Brooks, Robert E. Bruccoleri, Barry D. Olafson, David J. States, S. Swaminathan, and Martin Karplus. Charmm: A program for macromolecular energy, minimization, and dynamics calculations. *Journal of Computational Chemistry*, 4:187–217, 1983.

[26] Amedeo Caflisch, Stephan Fischer, and Martin Karplus. Docking by monte carlo minimization with a solvation correction: Application to an fkbp-substrate complex. *Journal of Computational Chemistry*, 18(6):723–743, 1997.

[27] Carlos J. Camacho, David W. Gatchell, S. Roy Kimura, and Sandor Vajda. Scoring docked conformations generated by rigid-body protein-protein docking. *Proteins: Structure, Function, and Genetics*, 40(3):525–537, July 2000.

[28] Adrian A. Canutescu, Andrew A. Shelenkov, and Jr. Roland L. Dunbrack. A graph-theory algorithm for rapid protein side-chain prediction. *Protein Science*, 12:2001–2014, 2003.

[29] Julio Castrillon-Candas, Vinay Siddavanahalli, and Chandrajit Bajaj. Nonequispaced fourier transforms for protein-protein docking. ICES Report 05-44, The University of Texas at Austin, Austin, TX, USA 78712, October 2005.

[30] Julio Castrillon-Candas, Vinay Siddavanahalli, and Chandrajit Bajaj. Nonequispaced fourier transforms for protein-protein docking. ICES Report 05-44, The University of Texas at Austin, Austin TX USA, October 2005.

[31] Chu-Fei Chang, Zhiyun Li, Amitabh Varshney, and Qiaode Jeffry Ge. Hierarchical image-based and polygon-based rendering for large-scale visualizations. *Scientific Visualization*, 2001.

[32] Rong Chen, Li Li, and Zhiping Weng. Zdock: An initial-stage protein-docking algorithm. *Proteins: Structure, Function, and Genetics, Special Issue: CAPRI - Critical Assessment of PRedicted Interactions . Issue Edited by Jol Janin*, 52(1):80–87, May 2003.

[33] Rong Chen and Zhiping Weng. A novel shape complementarity scoring function for protein-protein docking. *Proteins: Structure, Function, and Genetics*, 51(3):397–408, March 2003.

[34] Jacqueline Cherfils, Stephane Duquerroy, and Joel Janin. Protein-protein recognition analyzed by docking simulation. *Proteins: Structure, Function, and Genetics*, 11(4):271–280, 1991.

[35] Jack R. Collins, Stanley K. Burt, and John W. Erickson. Flap opening in hiv-1 protease simulated by 'activated' molecular dynamics. *Nature Structural Biology.*, 2(4):334–338, April 1995.

[36] Michael L. Connolly. Analytical molecular surface calculation. *Journal of Applied Crystallography*, 16:548–558, 1983.

[37] Michael L. Connolly. Solvent-accessible surfaces of proteins and nucleic acids. *Science*, 221(4612):709–713, 19 August 1983.

[38] Michael L. Connolly. Shape complementarity at the hemoglobin $\alpha_1\beta_1$ subunit interface. *Biopolymers*, 25(7):1229–1247, February 1986.

[39] Simona Cotesta, Fabrizio Giordanetto, Jean-Yves Trosset, Patrizia Crivori, Romano T. Kroemer, Pieter F.W. Stouten, and Anna Vulpetti. Virtual screening to enrich a compound collection with cdk2 inhibitors using docking, scoring, and composite scoring models. *Proteins: Structure, Function, and Bioinformatics*, 60(4):629–643, July 2005.

[40] Carolina Cruz-Neira, Randolph Langley, and Paul A. Bash. Vibe: A virtual biomolecular environment for interactive molecular modeling. *Computers & Chemistry*, 20(4):469–475, August 1996.

[41] Lucia Darsa, Bruno Costa, and Amitabh Varshney. Walkthroughs of complex environments using image-based simplification. *Computers and Graphics*, 22(1):25–34, January 1998.

[42] Paul E. Debevec, George Borshukov, and Yizhou Yu. Efficient view-dependent image-based rendering with projective texture-mapping. *In 9th Eurographics Rendering Workshop, Vienna, Austria*, June 1998.

[43] Warren L. DeLano. The pymol molecular graphics system. *World Wide Web http://www.pymol.org*, 2002.

[44] Renee L. DesJarlais, Robert P. Sheridan, J. Scott Dixon, Irwin D. Kuntz, and R. Venkataraghavan. Docking flexible ligands to macromolecular receptors by molecular shape. *Journal of Medicinal Chemistry*, 29(11):2149–2153, November 1986.

[45] Johan Desmet, Marc De Maeyer, Bart Hazes, and Ignace Lasters. The dead-end elimination theorem and its use in protein side-chain positioning. *Letters to Nature*, 356:539–542, April 1992.

[46] Huong Quynh Dinh, Ronald Metoyer, and Greg Turk. Real-time lighting changes for image-based rendering. *IASTED International Conference on Computer Graphics and Imaging*, 1998.

[47] Dina Duhovny, Ruth Nussinov, and Haim J. Wolfson. Efficient unbound docking of rigid molecules. In R. Guigo and D. Gusfield, editors, *Proceedings of the Fourth International Workshop on Algorithms in Bioinformatics*, pages 185–200, Springer-Verlag GmbH Rome, Italy, September 2002.

[48] Roland L. Dunbrack-Jr and Fred E. Cohen. Bayesian statistical analysis of protein side-chain rotamer preferences. *Protein Sciences*, 6:1661–1681, 1997.

[49] Bruce S. Duncan and Arthur J. Olson. Approximation and characterization of molecular surfaces. *Biopolymers*, 33(2):219–229, February 1993.

[50] Bruce S. Duncan and Arthur J. Olson. Shape analysis of molecular surfaces. *Biopolymers*, 33(2):231–238, February 1993.

[51] Herbert Edelsbrunner and Ernst P. Mucke. Three-dimensional alpha shapes. *ACM Transactions on Graphics*, 13(1):43–72, 1994.

[52] Adrian H. Elcock, David Sept, and J. Andrew McCammon. Computer simulation of protein-protein interactions. *Journal of Physical Chemistry B*, 105(8):1504–1518, February 2001.

[53] Todd J. A. Ewing and Irwin D. Kuntz. Critical evaluation of search algorithms for automated molecular docking and database screening. *Journal of Computational Chemistry*, 18(9):1175–1189, December 1998.

[54] Eran Eyal and Dan Halperin. Dynamic maintenance of molecular surfaces under conformational changes. In *SCG '05: Proceedings of the twenty-first annual symposium on Computational geometry*, pages 45–54, New York, NY, USA, 2005. ACM Press.

[55] Randima Fernando and Mark J. Kilgard. *The Cg Tutorial: The definitive guide to programmable real-time graphics*. Addison-Wesley Pub Co, February 2003.

[56] Juan Fernndez-Recio, Maxim Totrov, and Ruben Abagyan. Soft proteinprotein docking in internal coordinates. *Protein Science*, 11:280–291, 2002.

[57] Thomas E. Ferrin, Conrad C. Huang, L. E. Jarvis, and Robert Langridge. The midas display system. *Journal of Molecular Graphics*, 6:13–27, 1988.

[58] Daniel Fischer, Shuo Liang Lin, Haim L. Wolfson, and Ruth Nussinov. A geometry-based suite of moleculardocking processes. *Journal of Molecular Biology*, 248(2):459–477, 1995.

[59] Daniel Fischer, Raquel Norel, Ruth Nussinov, and Haim J. Wolfson. 3-d docking of protein molecules. In *CPM '93: Proceedings of the 4th Annual Symposium on Combinatorial Pattern Matching*, pages 20–34, London, UK, 1993. Springer-Verlag.

[60] Henry A. Gabb, Richard M. Jackson, and Michael J. E. Sternberg. Modelling protein docking using shape complementarity, electrostatics and biochemical information. *Journal of Molecular Biology*, 272(1):106–120, September 1997.

[61] Razif R. Gabdoulline and Rebecca C. Wade. Analytically defined surfaces to analyze molecular interaction properties. *J. of Molecular Graphics*, 14(6):341–353, December 1996.

[62] Michael Garland. Multiresolution modeling: Survey and future opportunities: State of the art report. *Eurographics*, pages 111–131, September 1999.

[63] Daniel K. Gehlhaar, Gennady Verkhivker, Paul A. Rejtoand David B. Fogel, Lawrence J. Fogel, and Stephan T. Freer. Docking conformationally flexible small molecules into a protein binding site through simulated evolution. In J.R. McDonnell, R.G. Reynolds, and D.B. Fogel, editors, *Evolutionary Programming IV: The Proc. of Fourth Annual Conference on Evolutionary Programming*, pages 615–627, MIT Press, Cambridge, MA, 1995.

[64] Mark Gerstein, Arthur M. Lesk, and Cyrus Chothia. Structural mechanisms for domain movements in proteins. *Biochemistry*, 33(22):6739–6749, June 1994.

[65] Nobuhiro Go. A theorem on amplitudes of thermal atomic fluctuations in large molecules assuming specific conformations calculated by normal mode analysis. *Biophysical Chemistry*, 35(1):105–112, January 1990.

[66] Andrew C. Good and W. Graham Richards. Rapid evaluation of shape similarity using gaussian functions. *Journal of Chemical Information and Computer Sciences*, 33(1):112–116, February 1993.

[67] David S. Goodsell. Molecular machinery: A tour of the protein data bank. Poster, http://www.rcsb.org/pdbstatic/education _discussion/molecule_of_the _month/poster_quickref.pdf.

[68] David S. Goodsell and Arthur J. Olson. Automated docking of substrates to proteins by simulated annealing. *Proteins:Structure, Function and Genetics*, 8(3):195–202, 1990.

[69] David Gotz, Ketan Mayer-Patel, and Dinesh Manocha. Irw: An incremental representation for image-based walkthroughs. *ACM Multimedia 2002*, 2002.

[70] J. Grant and B. Pickup. A gaussian description of molecular shape. *Journal of Physical Chemistry*, 99:3503–3510, 1995.

[71] Jonathan Greer and Bruce L. Bush. Macromolecular shape and surface maps by solvent exclusion. *Proceedings of the National Academy of Sciences of the United States of America*, 75(1):303–307, January 1978.

[72] Inbal Halperin, Buyong Ma, Haim Wolfson, and Ruth Nussinov. Principles of docking: An overview of search algorithms and a guide to scoring functions. *Proteins: Structure, Function and Genetics*, 47(4):409–443, June 2002.

[73] Andrew J. Hanson and Eric A. Wernert. Image-based rendering with occlusions via cubist images. In *Proceedings of the Conference on Visualization '98*, pages 327–334. IEEE Computer Society Press, 1998.

[74] Steven Hayward and Herman J. C. Berendsen. Systematic analysis of domain motions in proteins from conformational change; new results on

citrate synthase and t4 lysozyme. *Proteins, Structure, Function and Genetics*, 30(2):144–154, February 1998.

[75] Konrad Hinsen. Analysis of domain motions by approximate normal mode calculations. *Proteins: Structure, Function, and Genetics*, 33:417–429, 1998.

[76] Liisa Holm and Chris Sander. Parser for protein folding units. *Proteins*, 19(3):256–268, July 1994.

[77] William Humphrey, Andrew Dalke, and Klaus Schulten. VMD – Visual Molecular Dynamics. *Journal of Molecular Graphics*, 14:33–38, 1996.

[78] Wonpil Im, Dmitrii Beglov, and Benoit Roux. Continuum solvation model: electrostatic forces from numerical solutions to the poisson-boltzmann equation. *Computer Physics Communications*, 111:59–75, 1998.

[79] Wonpil Im, Michael S. Lee, and Charles L. Brooks III. Generalized born model with a simple smoothing function. *Journal of computational chemistry*, 24(14):1691–1702, November 2003.

[80] D. J. Jacobs and M. F. Thorpe. Generic rigidity percolation: The pebble game. *Phys. Rev. Lett.*, 75(22):4051–4054, Nov 1995.

[81] Donald J. Jacobs, Andrew J. Rader, Leslie A. Kuhn, and Michael F. Thorpe. Protein flexibility predictions using graph theory. *Proteins: Structure, Function, and Genetics*, 44:150–165, 2001.

[82] Joel Janin. Welcome to capri: A critical assessment of predicted interactions. *Proteins: Structure, Function, and Genetics*, 47(3):257, 2002.

[83] Stefan Jeschke and Michael Wimmer. Textured depth meshes for real-time rendering of arbitrary scenes. In *Proceedings of the 13th Eurographics workshop on Rendering*, pages 181–190. Eurographics Association, 2002.

[84] Fan Jianga and Sung-Hou Kim. "soft docking": Matching of molecular surface cubes. *Journal of Molecular Biology*, 219(1):79–102, May 1991.

[85] Gareth Jones, Peter Willett, and Robert C. Glen. Molecular recognition of receptor sites using a genetic algorithm with a description of desolvation. *Journal of Molecular Biology*, 245(1):43–53, January 1995.

[86] Gareth Jones, Peter Willett, Robert C. Glen, Andrew R. Leach, and Robin Taylor. Development and validation of a genetic algorithm for flexible docking. *Journal of Molecular Biology*, 267(3):727–748, April 1997.

[87] Tao Ju, Frank Losasso, Scott Schaefer, and Joe Warren. Dual contouring of hermite data. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 339–346, New York, NY, USA, 2002. ACM Press.

[88] Richard S. Judson, E. P. Jaeger, and Adi M. Treasurywala. A genetic algorithm based method for docking flexible molecules. *Journal of*

Molecular Structure: THEOCHEM, 308:191–206, May 1994.

[89] Wolfgang Kabsch. A discussion of the solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A*, 34(5):827–828, September 1978.

[90] Laxmikant Kale, Robert Skeel, Milind Bhandarkar, Robert Brunner, Attila Gursoy, Neal Krawetz, James Phillips, Artiomo Shinozaki, Krishnan Varadarajan, and Klaus Schulten. Namd2: greater scalability for parallel molecular dynamics. *Journal of Computational Physics*, 151(1):283–312, 1999.

[91] Ephraim Katchalski-Katzir, Isaac Shariv, Miriam Eisenstein, Asher A. Friesem, Claude Aflalo, and Ilya A. Vakser. Molecular surface recognition: determination of geometric fit between proteins and their ligands by correlation techniques. *Proceedings of the National Academy of Sciences of the United States of America*, 89(6):2195–2199, March 1992.

[92] Ozlem Keskin, Robert L. Jernigan, and Ivet Bahar. Proteins with similar architecture exhibit similar large-scale dynamic behavior. *Biophysical Journal*, 78(4):2093–2106, April 2000.

[93] Julio A. Kovacs, Pablo Chacn, Yao Cong, Essam Metwally, and Willy Wriggers. Fast rotational matching of rigid bodies by fast fourier transform acceleration of five degrees of freedom. *Acta Crystallographica, Biological Crystallography*, D59(8):1371–1376, August 2003.

[94] Julio A. Kovacs and Willy Wriggers. Fast rotational matching. *Acta Crystallographica, Biological Crystallography*, D58(8):1282–1286, August 2002.

[95] Marc Van Kreveld, Mark Overmars, Otfried Schwarzkopf, and Mark De Berg. *Computational geometry: Algorithms and applications*. Springer-Verlag Telos, 1997.

[96] Romano T. Kroemer, Anna Vulpetti, Joseph J. McDonald, Douglas G. Rohrer, Jean-Yves Trosset, Fabrizio Giordanetto, Simona Cotesta, Colin McMartin, Mats Kihln, and Pieter F. W. Stouten. Assessment of docking poses: interactions-based accuracy classification (ibac) versus crystal structure deviations. *Journal of Chemical Information and Computer Sciences*, 44:871–888, 2004.

[97] James J. Kuffner. Effective sampling and distance metrics for 3d rigid body path planning. *IEEE International Conference on Robotics and Automation*, May 2004.

[98] F. S. Kuhl, Gorden M. Crippen, and Donald K. Friesen. A combinatorial algorithm for calculating ligand binding. *Journal of Computational Chemistry*, 5(1):24–34, 1984.

[99] Irwin D. Kuntz, Jeffrey M. Blaney, Stuart J. Oatley, Robert Langridge, and Thomas E. Ferrin. A geometric approach to macromolecule-ligand interactions. *Journal of Molecular Biology*, 161(2):269–288, October 1982.

[100] Yehezkel Lamdan and Haim J. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. In *Second International Conference on Computer Vision*, pages 238–249, 1988.

[101] Eaton E. Lattman. Optimal sampling of the rotation function. *Acta Crystallographica Section B*, 28(4):1065–1068, April 1972.

[102] Andrew R. Leach. Ligand docking to proteins with discrete side-chain flexibility. *Journal of Molecular Biology*, 235(1):345–356, January 1994.

[103] Andrew R. Leach. *Molecular modelling: Principles and applications*. Addison Wesley, 1996.

[104] Andrew R. Leach and Irwin D. Kuntz. Conformational analysis of flexible ligands in macromolecular receptor sites. *Journal of Computational Chemistry*, 13(6):730–748, 1992.

[105] B. Lee and Frederic M. Richards. The interpretation of protein structures: estimation of static accessibility. *Journal of Molecular Biology*, 55(3):379–400, February 1971.

[106] Dong Hoon Lee and Soon Ki Jung. Capture configuration for image-based street walkthroughs. *International Conference on Cyberworlds*, 03–05:151, December 2003.

[107] Jonathan Leech, Jan F. Prins, and Jan Hermans. Smd: Visual steering of molecular dynamics for protein design. *IEEE Computational Science and Engineering*, 3(4):38–45, Winter 1996.

[108] Hans-Peter Lenhof. New contact measures for the protein docking problem. In *RECOMB '97: Proceedings of the first annual international conference on Computational molecular biology*, pages 182–191, New York, NY, USA, 1997. ACM Press.

[109] David Levine, Michael Facello, Philip Hallstrom, Gregory Reeder, Brian Walenz, and Fred Stevens. Stalk: An interactive system for virtual molecular docking. *IEEE Computational Science and Engineering*, 04(2):55–65, 1997.

[110] Michael Levitt, Miriam Hirshberg, Ruth Sharon, and Valerie Daggett. Potential energy function and parameters for simulations of the molecular dynamics of proteins and nucleic acids in solution. *Computer Physics Communications*, 91:215–231, 1995.

[111] Adriano Lopes and Ken Brodlie. Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. *IEEE transactions on visualization and computer graphics*, 9(1):16–29, 2003.

[112] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, volume 21, pages 163–169, New York, NY, USA, July 1987. ACM Press.

[113] Jianpeng Ma and Martin Karplus. The allosteric mechanism of the

chaperonin groel: A dynamic analysis. *Proceedings of the National Academy of Sciences USA.*, 95(15):8502–8507, July 1998.

[114] Ulrika Magnusson, Barnali Neel Chaudhuri, Junsang Ko, Chankyu Park, T. Alwyn Jones, and Sherry L. Mowbray. Hinge-bending motion of d-allose binding protein from escherichia coli: three open conformations. *Journal of Biological Chemistry*, 277(16):14077–14084, April 2002.

[115] Vladimir N. Maiorov and Ruben A. Abagyan. A new method for modeling large-scale rearrangements of protein domains. *Proteins*, 27:410–424, 1997.

[116] Jeffrey G. Mandell1, Victoria A. Roberts, Michael E. Pique, Vladimir Kotlovyi, Julie C. Mitchell, Erik Nelson, Igor Tsigelny, and Lynn F. Ten Eyck. Protein docking using continuum electrostatics and geometric fit. *Protein Engineering*, 14(2):105–113, February 2001.

[117] Eric Martz. Protein explorer: Easy yet powerful macromolecular visualization. *Trends in Biochemical Sciences*, 27:107–109, February 2002. http://proteinexplorer.org.

[118] Nelson Max. Computer representation of molecular surfaces. *IEEE Computer Graphics and Applications*, pages 21–29, August 1983.

[119] Colin McMartin and Regine S. Bohacek. Qxp: powerful, rapid computer algorithms for structure-based drug design. *Journal of Computer-Aided Molecular Design*, 11:333–344, 1997.

[120] Michael Meyer, Peter Wilson, and Dietmar Schomburg. Hydrogen bonding and molecular surface shape complementarity as a basis for protein docking. *Journal of Molecular Biology*, 264(1):199–210, November 1996.

[121] Paul G. Mezey. The shape of molecular charge distributions: Group theory without symmetry. *Journal of Computational Chemistry*, 8(4):462–469, June 1987.

[122] Paul G. Mezey. Shape group studies of molecular similarity: Shape groups and shape graphs of molecular contour surfaces. *Journal Journal of Mathematical Chemistry*, 2(4):299–323, October 1988.

[123] Paul G. Mezey. *Shape in Chemistry; An introduction to molecular shape and topology*. VCH Inc, 1993.

[124] Miho Yamada Mizutani, Nobuo Tomioka, and Akiko Itai. Rational automatic search method for stable docking models of protein and ligand. *Journal of Molecular Biology*, 243(2):310–326, October 1994.

[125] Ral Mndez, Raphal Leplae, Leonardo De Maria, and Shoshana J. Wodak. Assessment of blind predictions of protein-protein interactions: Current status of docking methods. *Proteins: Structure, Function, and Genetics*, 52(1):51–67, 2003.

[126] Gidon Moont, Henry A. Gabb, and Michael J.E. Sternberg. Use of pair potentials across protein interfaces in screening predicted docked

complexes. *Proteins: Structure, Function, and Genetics*, 35(3):364–373, September 1999.

[127] Garrett M. Morris, David S. Goodsell, Robert S. Halliday, Ruth Huey, William E. Hart, Richard K. Belew, and Arthur J. Olson. Automated docking using a lamarckian genetic algorithm and an empirical binding free energy function. *Journal of Computational Chemistry*, 19(14):1639–1662, 1998.

[128] William L. Nichols, George D. Rose, Lynn F. Ten Eyck, and Bruno H. Zimm. Rigid domains in proteins: an algorithmic approach to their identification. *Proteins*, 23(1):38–48, September 1995.

[129] Alfredo Di Nola, Danilo Roccatano, and Herman J. C. Berendsen. Molecular dynamics simulation of the docking of substrates to proteins. *Proteins*, 19(3):174–182, July 1994.

[130] Raquel Norel, Daniel Fischer, Haim J. Wolfson, and Ruth Nussinov. Molecular surface recognition by a computer vision-based technique. *Protein engineering*, 7(1):39–46, January 1994.

[131] Raquel Norel, Shuo L. Lin, Haim J. Wolfson, and Ruth Nussinov. Molecular surface complementarity at protein-protein interfaces: The critical role played by surface normals at well placed, sparse, points in docking. *Journal of Molecular Biology*, 252(2):263–273, September 1995.

[132] Connie M. Oshiro, Irwin D. Kuntz, and J. Scott Dixon. Flexible ligand docking using a genetic algorithm. *Journal of Computer-aided Molecular Design*, 9(2):113–130, April 1995.

[133] Sanghun Park, Chandrajit Bajaj, and Vinay Siddavanahalli. Interactive rendering of adaptive mesh refinement data. *Proceedings of IEEE Visualization, Boston, MA*, pages 521–524, 2002.

[134] Jay W. Ponder and Frederic M. Richards. Tertiary templates for proteins: Use of packing criteria in the enumeration of allowed sequences for different structural classes. *Journal of Molecular Biology*, 193:775–791, 1987.

[135] Daniel Potts and Gabriele Steidl. Fast summation at nonequispaced knots by nffts. *SIAM Journal on Scientific Computing*, 24(6):2013–2037, 2003.

[136] Daniel Potts, Gabriele Steidl, and Manfred Tasche. *Fast fourier transform for nonequispaced data: A tutorial, in Modern Sampling Theory: Mathematics and Applications*, chapter 12, pages 249–274. 1998.

[137] Catherine Lamb Propst and Thomas J. Perun. *Computer-aided drug design : methods and applications*. New York : Marcel Dekker, 1989.

[138] George D. Purvis and Chris Culberson. On the graphical display of molecular electrostatic force-fields and gradients of the electron density. *Journal of Molecular Graphics*, June 1986.

[139] Matthias Rarey, Bernd Kramer, and Thomas J. Lengauer. Multiple automatic base selection: Protein-ligand docking based on incremental construction without manual intervention. *Journal of Computer-Aided Molecular Design*, 11:369–384, 1997.

[140] Matthias Rarey, Bernd Kramer, Thomas J. Lengauer, and Gerhard Klebe. A fast flexible docking method using an incremental construction algorithm. *Journal of Molecular Biology*, 261(3):470–489, August 1996.

[141] Frederic M. Richards. Areas, volumes, packing, and protein structure. *Annual Review of Biophysics and Bioengineering*, 6:151–176, June 1977.

[142] David W. Ritchie. *Parametric Protein Shape Recognition*. Phd thesis, Departments of Computer Science & Molecular and Cell Biology, University of Aberdeen, King's College, Aberdeen, UK, September 1998.

[143] David W. Ritchie. Evaluation of protein docking predictions using hex 3.1 in capri rounds 1 and 2. *Proteins: Structure, Function, and Genetics*, 52(1):98–106, July 2003.

[144] David W. Ritchie and Graham J.L. Kemp. Protein docking using spherical polar fourier correlations. *Proteins: Structure, Function, and Genetics*, 39(2):178–194, March 2000.

[145] Bilha Sandak, Ruth Nussinov, and Haim J. Wolfson. An automated computer vision and robotics-based technique for 3-d flexible biomolec-

ular docking and matching. *Computer Applications in the Biosciences : CABIOS*, 11(1):87–99, February 1995.

[146] Michel Sanner, Arthur Olson, and Jean-Claude Spehner. Reduced surface: an efficient way to compute molecular surfaces. *Biopolymers*, 38(3):305–320, March 1996.

[147] Michel F. Sanner, Arthur J. Olson, and Jean-Claude Spehner. Fast and robust computation of molecular surfaces. In *Proceedings of the eleventh annual symposium on Computational geometry*, pages 406–407. ACM Press, 1995.

[148] Roger Sayle and E. James Milner-White. Rasmol: Biomolecular graphics for all. *Trends in Biochemical Sciences (TIBS)*, 20(9):374, September 1995.

[149] Gernot Schaufler and Wolfgang Sturzlinger. A three dimensional image cache for virtual reality. *Computer Graphics Forum (Eurographics '96)*, 15(3):227–235, 1996.

[150] Raj Shekhar, Elias Fayyad, Roni Yagel, and J. Fredrick Cornhill. Octree-based decimation of marching cubes surfaces. *IEEE Visualization*, pages 335–342, 1996.

[151] Craig M. Shepherd, Ian A. Borelli, Gabriel Lander, Padmaja Natarajan, Vinay Siddavanahalli, Chandrajit Bajaj, John E. Johnson, Charles

L. Brooks III, and Vijay S. Reddy. Viperdb: a relational database for structural virology. *Database issue of Nucleic Acid Research*, 34, 2006.

[152] Brian. K. Shoichet, D. L. Bodian, and Irwin D. Kuntz. Molecular docking using shape descriptors. *Journal of Computational Chemistry*, 13(3):380–397, 1992.

[153] Brian K. Shoichet and Irwin D. Kuntz. Protein docking and complementarity. *Journal of Molecular Biology*, 221(1):327–346, September 1991.

[154] Vinay Siddavanahalli and Chandrajit Bajaj. Fast error-bounded surfaces and derivatives computation for volumetric particle data. CS Technical report TR-06-06, The University of Texas at Austin, Austin, Texas, USA 78712, December 2005.

[155] Asim S. Siddiqui and Geoffrey J. Barton. Continuous and discontinuous domains: an algorithm for the automatic generation of reliable protein domain definitions. *Protein Science*, 4(5):872–884, May 1995.

[156] Robert D. Skeel, Ismail Tezcan, and David J. Hardy. Multiple grid methods for classical molecular dynamics. *Journal of Computatioanl Chemistry*, 23(6):673–684, 2002.

[157] Bong-Soo Sohn, Chandrajit Bajaj, and Vinay Siddavanahalli. Feature based volumetric video compression for interactive playback. *Pro-*

ceedings of IEEE/SIGGRAPH Symposium on Volume Visualization and Graphics, Boston, MA, pages 89–96, 2002.

[158] Bong-Soo Sohn, Chandrajit Bajaj, and Vinay Siddavanahalli. Volumetric video compression for interactive playback. *Computer Vision and Image Understanding*, 96:435–452, 2004.

[159] W. Clark Still, Anna Tempczyk, Ronald C. Hawley, and Thomas Hendrickson. Semianalytical treatment of solvation for molecular mechanics and dynamics. *Journal of American Chemical Society*, 112(16):6127–6129, 1990.

[160] John E. Stone, Justin Gullingsrud, and Klaus Schulten. A system for interactive molecular dynamics simulation. In *Proceedings of the 2001 symposium on Interactive 3D graphics*, pages 191–194. ACM Press, 2001.

[161] Florence Tama, Florent Xavier Gadea, Osni Marques, and Yves-Henri Sanejouand. Building-block approach for determining low-frequency normal modes of macromolecules. *Proteins*, 41(1):1–7, October 2000.

[162] Miguel Teodoro, Jr. George N. Phillips, and Lydia E. Kavraki. Molecular docking: A problem with thousands of degrees of freedom. *IEEE International Conference on Robotics and Automation, Seoul, Korea*, 2001.

[163] Miguel L. Teodoro, George N. Phillips, Jr., and Lydia E. Kavraki. A dimensionality reduction approach to modeling protein flexibility. In *Proceedings of the sixth annual international conference on Computational biology*, pages 299–308. ACM Press, 2002.

[164] Amitabh Varshney and Jr. Frederick P. Brooks. Fast analytical computation of richards's smooth molecular surface. In *VIS '93: Proc. 4th conf. on Vis. '93*, pages 300–307, 1993.

[165] R. Voorintholt, M. T. Kosters, G. Vegter, G. Vriend, and W. G. Hol. A very fast program for visualizing protein surfaces, channels and cavities. *Journal of Molecular Graphics*, 7(4):243–245, December 1989.

[166] P. Duane Walker, Gustavo A. Arteca, and Paul G. Mezey. A complete shape characterization for molecular charge densities represented by gaussian-type functions. *Journal of Computational Chemistry*, 33(2):231–238, February 1991.

[167] Peter H. Walls and Michael J. E. Sternberg. New algorithm to model protein-protein recognition based on surface complementarity. applications to antibody-antigen docking. *Journal of Molecular Biology*, 228(1):277–297, November 1992.

[168] Huajun Wang. Grid-search molecular accessible surface algorithm for solving the protein docking problem. *Journal of Computational Chemistry*, 12(6):746–750, 1991.

[169] Jian Wang, Peter A. Kollman, and Irwin D. Kuntz. Flexible ligand docking: A multistep strategy approach. *Proteins: Structure, Function, and Genetics*, 36(1):1–19, October 1999.

[170] J. Andre C. Weideman. Computation of the complex error function. *SIAM Journal on Numerical Analysis*, 31(5):1497–1518, 1994.

[171] William Welch, Jim Ruppert, and Ajay N. Jain. Hammerhead: fast, fully automated docking of flexible ligands to protein binding sites. *Chemistry & Biology*, 3(6):449–462, June 1996.

[172] John Westbrook and Paula M. Fitzgerald. *Structural Bioinformatics*, pages 161–179. P. E. Bourne and H. Weissig, John Wiley & Sons, Inc., 2003.

[173] Rudiger Westermann, Leif Kobbelt, and Thomas Ertl. Real-time exploration of regular volume data by adaptive reconstruction of iso-surfaces. *The Visual Computer*, pages 100–111, 1999.

[174] Willy Wriggers and Pablo Chacon. Modeling tricks and fitting techniques for multiresolution structures. *Structure*, 9(9):779–788, September 2001.

[175] Willy Wriggers and Klaus Schulten. Protein domain movements: Detection of rigid domains and visualization of effective rotations in comparisons of atomic coordinates. *Proteins: Structure, Function, and Genetics*, 29:1–14, 1997.

[176] Jinbo Xu and Bonnie Berger. Fast and accurate algorithms for protein side-chain packing. *J. ACM*, 53(4):533–557, 2006.

[177] Anna Yershova and Steven M. LaValle. Deterministic sampling methods for spheres and so(3). *IEEE International Conference on Robotics and Automation*, 4:3974–3980, 2004.

[178] Tony You and Donald Bashford. An analytical algorithm for the rapid determination of the solvent accessibility of points in a three-dimensional lattice around a solute molecule. *Journal of Computational Chemistry*, 16(6):743–757, 1995.

[179] Shi-Yi Yue. Distance-constrained molecular docking by simulated annealing. *Protein engineering.*, 4(2):177–184, December 1990.

[180] Maria I. Zavodszky and Leslie A. Kuhn. Side-chain flexibility in proteinligand binding: The minimal rotation hypothesis. *Protein Science*, 14:1104–1114, 2005.

[181] Micheal H. Zehfus and George D. Rose. Compact units in proteins. *Biochemistry*, 25(19):5759–5765, September 1986.

[182] Yong Zhao, Daniel Stoffler, and Michel Sanner. Hierarchical and multiresolution representation of protein flexibility. *Bioinformatics*, 22(22):2768–2774, 2006.

Index

Vita

Vinay Kiranshankar Siddavanahalli was born in Bangalore, India on January 10, 1978. His parents are Parvati and Kiranshankar Nijalingappa. He obtained a Bachelor of Engineering degree in the Department of Computer Sciences in Sri Jayachamarajendra College of Engineering in Mysore, 2000 and also worked for a year on 'MPEG text detection and recognition', as part of his thesis in The Indian Institute of Science, Bangalore, India, under Dr Ramakrishnan. He joined the Department of Computer Sciences at The University of Texas at Austin in August 2000. He also received a Masters of Science from the same university in 2006. Current research interests include Computer Graphics, Computational Geometry and Computational Biology.

Permanent address: 4501 Speedway, 102
Austin, Texas 78751

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.